

Generalizable VLA Finetuning via Representation Anchoring and Language-Action Alignment

Dwip Dalal¹, Shivansh Patel¹, Chahit Jain¹, Jeonghwan Kim¹, Utkarsh Mishra², Alex Baratian¹, Hyeonjeong Ha¹, Heng Ji¹, Svetlana Lazebnik^{1*}, Unnat Jain^{3*}

¹University of Illinois Urbana-Champaign ²Texas A&M University

³University of California, Irvine

Abstract:

Finetuning a pretrained vision-language model (VLM) on robot demonstrations via behavior cloning (BC) has become the standard recipe for vision-language-action (VLA) policies. However, BC finetuning progressively overwrites the pretrained representations that support visual and semantic generalization. Co-training on web image-text data, a common remedy, does not prevent this; it applies language and action losses to separate observations, leaving VLAs with language-action misalignment that standard manipulation benchmarks do not expose. We propose Anchor-Align, which augments BC with two objectives: *Vision-Language Anchoring* distills layer-wise representations from a frozen VLM copy to prevent this drift, while *Language-Action Alignment* converts each action target into a discrete motion-direction label and jointly trains language and action prediction on the same robot observation. On a physical xArm7 robot, across two widely used VLA architectures, Anchor-Align improves real-robot success on both (28% $\rightarrow$ 54% and 37% $\rightarrow$ 60%). At scale in simulation, we demonstrate consistent improvements on OOD perturbations, perceptual robustness, and long-horizon control across LIBERO-PRO, LIBERO-Plus, and CALVIN, respectively, suggesting that preserving pretrained representations and effective action learning are not fundamentally at odds. Project page: anchoralignvla.github.io

Keywords: Vision-Language-Action Models, Robot Manipulation, Catastrophic Forgetting, Language-Action Alignment, Out-of-Distribution Generalization

1 Introduction

Vision-Language-Action (VLA) models have become a popular approach for learning robot manipulation policies [3, 4, 6, 18, 20, 36, 37, 40, 43, 57, 61, 70, 72, 73]. VLAs are typically trained by finetuning a pretrained vision-language model (VLM) on expert demonstrations via supervised action prediction, known as behavior cloning (BC). This can be done through direct regression [37, 72] or flow-matching and diffusion [3, 4, 69]. The premise is that such finetuning should transfer the VLM’s semantic priors (understanding of spatial layout, directions, color, shape, etc.) to the resulting control policy.

Consider a VLA policy finetuned to “pick up the green mug and place it on the plate” in a scene containing both a green and pink mug (Fig. 1). Since the two mugs share the same shape and manipulation affordances, the policy should generalize to “pick up the pink mug” if finetuning preserves the VLM’s color understanding. However, with existing methods, it does not. In fact, BC finetuning corrupts the very prior that makes VLMs worth adapting, leading to two failure modes that persist even under good training practices.

First, standard BC optimizes only the action prediction loss, with nothing protecting the VLM’s pretrained representations from being overwritten. Over the course of finetuning, these updates progressively erase the visuolinguistic and spatial concepts the VLM acquired during internet-scale

*Equal advising

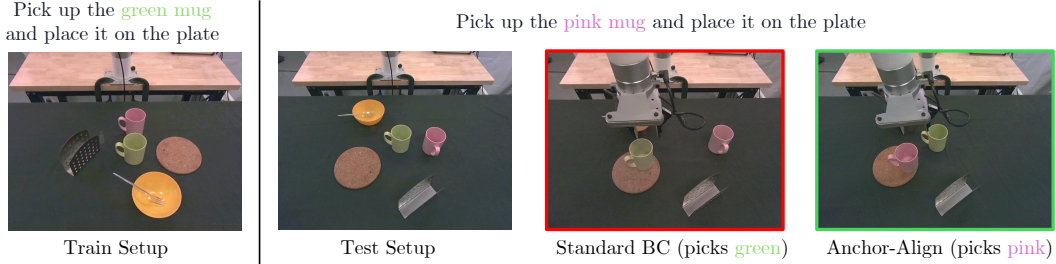


Figure 1: **Out-of-distribution test for VLAs.** Standard BC, finetuned to pick up the green mug, reaches for the green mug even when instructed to pick up the pink mug, whereas Anchor-Align retains the VLM’s representations and successfully completes the task.

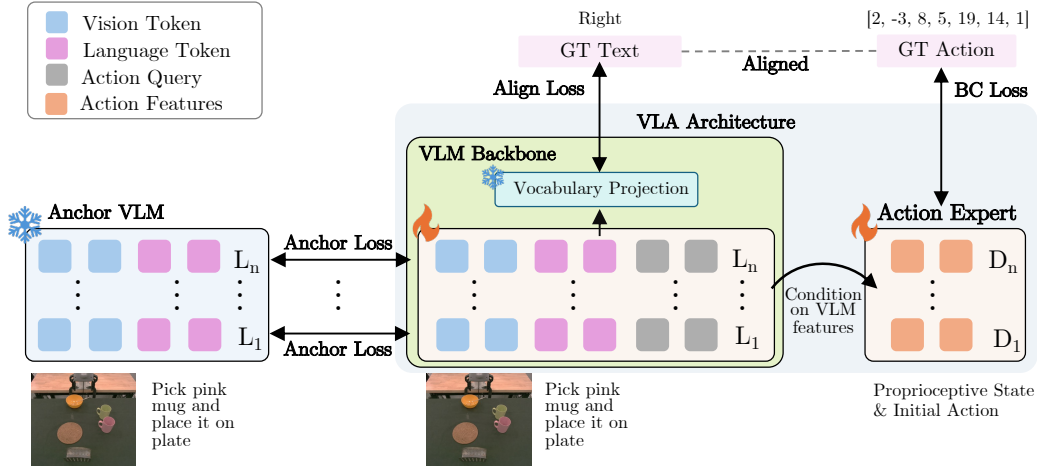


Figure 2: **Anchor-Align preserves VLM priors and aligns them with action during VLA finetuning.** Vision-Language Anchoring distills representations from a frozen pretrained VLM into the trainable VLA at every transformer layer (Anchor loss), preventing catastrophic forgetting. Language-Action Alignment programmatically converts each ground-truth demonstration trajectory into a discrete language label and trains the model to predict this label on the same observation it acts on (Align loss). Together, these objectives retain the pretrained representations while grounding action generation in VLM semantics.

pretraining. Concretely, on our physical xArm7 robot, a VLA trained to pick up the green mug reaches for the green mug 90% of the time even when instructed to pick up the pink one, indicating that color grounding has been erased from the backbone. On LIBERO-PRO’s position-swap test, where object positions are rearranged at evaluation, the policy almost always replays its memorized training trajectory rather than acting on the current observation. To fix this failure mode, recent work has proposed co-training, or augmenting continuous action data with general-purpose VQA, captioning, and scene-description data [21, 40, 76, 82] to prevent catastrophic forgetting. Yet we find that on the position-swap test, co-trained VLAs score 0%. Evidently, losses on such auxiliary data do not place a sufficient constraint on the backbone’s representations. As a more direct remedy, we propose *Vision-Language Anchoring*, or maintaining a frozen copy of the pretrained VLM (Fig. 2, left) and applying a distillation loss throughout finetuning. This requires no additional data or architectural changes, yet recovers out-of-distribution (OOD) generalization.

The second failure mode runs even deeper. Co-training gives the VLA’s two heads disjoint supervision: the action head is supervised by expert demonstrations while the language head is supervised by losses on generic image-text tasks. Because the heads share a backbone but are never supervised on the same robot observation, their predictions may contradict each other: when the action head predicts a movement to the right, the language head may predict “left.” To fix this misalignment and further improve task success, we introduce *Language-Action Alignment*, which co-supervises both heads on the same robot observation during finetuning (Fig. 2, top). To supply the aligned supervisory data, we automatically convert continuous action targets in expert demonstrations into discrete action directions expressed in language.

We propose combining Vision-Language Anchoring and Language-Action Alignment with the standard BC loss to form **Anchor-Align**, a VLA finetuning method with consistent improvements across simulation and real-world experiments under both regression and flow-matching action heads. In simulation, our method improves OOD generalization on LIBERO-PRO and LIBERO-Plus benchmarks, which introduce unseen spatial rearrangements, camera perturbations and other shifts absent from training; and on the long-horizon CALVIN benchmark. In real-world experiments on an xArm7 robot, Anchor-Align improves performance under unseen spatial rearrangements, semantic perturbations, and cluttered scenes. Beyond these results, our Language-Action Alignment framework provides the first direct diagnosis of language-action misalignment in co-trained VLAs and shows that better alignment improves action accuracy.

2 Related Works

General-Purpose VLAs for Robot Manipulation. VLAs use pretrained VLMs as foundation policies for generalist manipulation, adapting to downstream tasks with minimal finetuning [6, 18, 20, 37, 43, 57, 61, 70]. RT-2 [83] co-finetunes a VLM on robot trajectories, showing web-scale pretraining yields emergent generalization to novel objects and compositional reasoning, while OpenVLA [36] scales this via cross-embodiment training on Open X-Embodiment [53]. Architectural variants include flow-matching continuous actions ($\pi_0, \pi_{0.5}$) [4, 21], a frozen-VLM + diffusion-transformer dual system for humanoids [3], and a billion-parameter diffusion expert with curriculum learning [73]. VLA-Adapter [72] instead avoids large-scale robotic pretraining, using a lightweight Bridge Attention mechanism to turn any off-the-shelf VLM into a strong VLA. StarVLA [69] likewise forgoes robotic pretraining, offering a modular framework that pairs an arbitrary VLM backbone with an arbitrary action head. Unlike these, we investigate which finetuning recipe best preserves the VLM’s representations. We build on VLA-Adapter and StarVLA as our base architectures: both finetune a pretrained VLM directly on expert demonstrations, and together they cover both regression and flow-matching action heads.

Preserving VLM Representations in VLAs. Pretrained visual representations are a well-established driver of robot manipulation performance [1, 17, 52, 58, 59, 71]. In VLAs, this requirement extends beyond the vision encoder to the full VLM backbone, since semantic and spatial reasoning are distributed across both vision and language representations. Prior VLAs commonly mitigate catastrophic forgetting by co-training on robot and generic vision-language data [40, 56, 73, 76], with variants adding action-grounding objectives, perception tokens, or chain-of-thought plans [40, 76, 78, 81, 82]. Grover et al. [27] instead preserve pretrained features architecturally, anchoring a frozen vision encoder with a trainable one. MAPS [30] penalizes each module’s weights for drifting from the pretrained checkpoint, constraining vision layers more tightly than action-oriented ones, and Kachaev et al. [34] align visual patch features at a single intermediate layer with a frozen vision foundation model. Our anchoring differs in both target and scope: it distills the hidden states of vision and text tokens at every decoder layer from a frozen copy of the pretrained VLM itself, directly preserving the backbone’s vision-language representations. Additionally, unlike Anchor-Align, none of these methods aligns language and action on the same robot observation.

Embodied Question Answering. Embodied QA has long tested language understanding under physical grounding, from navigation-centric QA [13, 16] to foundation-model reasoning [14, 15, 50] and robot-manipulation QA [10, 24, 56]. Yet a critical gap persists for VLAs: co-trained VLAs [76, 82, 83] evaluate language understanding on generic benchmarks [31, 67], and never test whether the model’s language predictions are consistent with its action predictions on the same observation.

3 Anchor-Align Method

This section presents Anchor-Align (Fig. 2), a recipe that finetunes a pretrained VLM into a continuous action policy while preserving its pretrained VL representations. We first describe our base VLA architecture and training objective (Sec. 3.1), then the two objectives Anchor-Align adds on top of standard BC: *Vision-Language Anchoring* (Sec. 3.2) and *Language-Action Alignment* (Sec. 3.3).

3.1 Base VLA Architecture

We consider VLAs that adopt a pretrained vision-language backbone with a dedicated action head [69, 72], as shown in Figure 2. The backbone encodes the visual observation and language instruction into a hierarchy of multimodal representations. Separately, robot proprioception is encoded by a lightweight state projector and provided to the action pathway as a conditioning signal. A fixed set of learned action queries extracts task-relevant information from the vision-language representations, either through joint self-attention within the backbone or through cross-attention in the action head. The action head then combines the resulting query features with the encoded robot state to predict a fixed-horizon chunk of continuous 7-DoF actions (3D translation and rotation offsets plus gripper control) using either direct regression or conditional flow matching.

We train this architecture with three objectives. The primary term is the standard BC loss $\mathcal{L}_{\text{action}}$ between the predicted and ground-truth action chunks, $\hat{\mathbf{a}}^{\text{cont}}$ and $\mathbf{a}^{\text{cont}}$ respectively. The form of the BC loss depends on the action-head architecture: an L_1 loss for a regression action head, and an L_2 loss over the predicted velocity for a flow-matching action head. To it we add two objectives, both evaluated on the same robot observation: an anchoring loss $\mathcal{L}_{\text{anchor}}$ that preserves the backbone’s pretrained vision-language representations (Sec. 3.2), and an alignment loss $\mathcal{L}_{\text{align}}$ that grounds the model’s language predictions in its actions (Sec. 3.3). The three terms are optimized jointly:

$$\mathcal{L}_{\text{total}} = \mathcal{L}_{\text{action}} + \lambda_{\text{anchor}} \mathcal{L}_{\text{anchor}} + \lambda_{\text{align}} \mathcal{L}_{\text{align}}, \quad (1)$$

where λ_{anchor} and λ_{align} weight the two added objectives against the action loss.

3.2 Vision-Language Anchoring

To prevent catastrophic forgetting, we anchor the VLA’s backbone VLM with a frozen copy of the same VLM (the *anchor* VLM), which processes the same input batch (images, text) in parallel. This distills the original pretrained VLM’s representations into the trainable backbone. In effect, the anchor VLM serves as a frozen teacher, a role well established in the ML literature: feature distillation trains a student to match a teacher’s representations [28, 33, 64], and anti-forgetting methods regularize a finetuned model toward its pretrained copy in weight space [38, 42] or representation space [19, 41, 51]. We anchor hidden states of the vision and text tokens between the backbone and anchor VLM at every decoder layer $u \in \mathcal{D}$, where $\mathcal{D}$ is the set of decoder layers. Let $\mathbf{m}$ denote the set of vision and text positions. The per-layer anchoring loss is

$$\mathcal{L}_{\text{anchor}}^{(u)} = \|\mathbf{H}_u^S[\mathbf{m}] - \mathbf{H}_u^A[\mathbf{m}]\|_F^2, \quad (2)$$

where $\mathbf{H}_u^S \in \mathbb{R}^{N \times d}$ and $\mathbf{H}_u^A \in \mathbb{R}^{N \times d}$ are the hidden states of the trainable backbone (S) and the anchor (A) at decoder layer u , with N the number of vision and text tokens and d the hidden dimension of the VLM. $\|\cdot\|_F^2$ is the sum of squared element-wise differences (squared Frobenius norm). The total anchoring loss averages over all $|\mathcal{D}|$ decoder layers:

$$\mathcal{L}_{\text{anchor}} = \frac{1}{|\mathcal{D}|} \sum_{u \in \mathcal{D}} \mathcal{L}_{\text{anchor}}^{(u)}. \quad (3)$$

3.3 Language-Action Alignment

As illustrated in Fig. 3, we programmatically convert the continuous action target $\mathbf{a}^{\text{cont}}$ into a motion-direction label $\mathbf{a}^{\text{lang}}$ for each non-stationary demonstration chunk and use this label to supervise the VLM backbone on the same observation on which the action head predicts the continuous actions. Concretely, we use the last-layer hidden state of the last instruction token, immediately before action prediction. This state summarizes the visual prefix and language instruction, making it a natural context representation for predicting the robot’s next motion direction. With N vision and text tokens, this is the N -th (last) token, so this pre-action hidden state is $\mathbf{h}^{\text{pre}} = \mathbf{H}^S[N] \in \mathbb{R}^d$, where $\mathbf{H}^S$ is the backbone VLM’s last-layer hidden state. We then project $\mathbf{h}^{\text{pre}}$ onto the vocabulary through a learned projection and the frozen pretrained language head:

$$\hat{\mathbf{a}}^{\text{lang}} = \mathbf{W}_{\text{lm}} \mathbf{W}_{\text{proj}} \mathbf{h}^{\text{pre}} \in \mathbb{R}^{|\mathcal{V}|}, \quad (4)$$

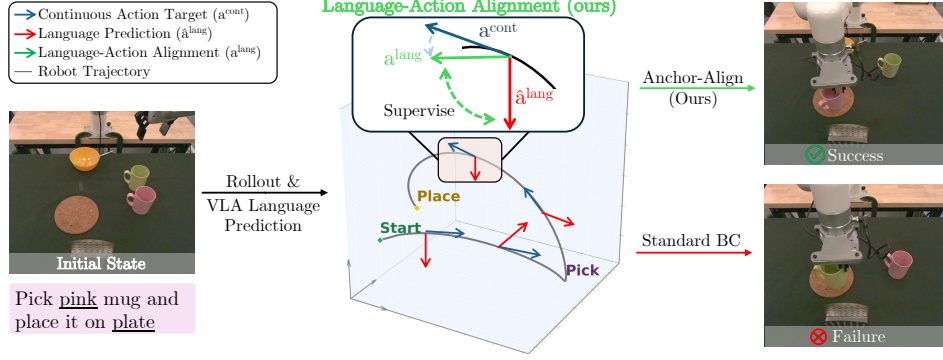


Figure 3: **Anchor-Align VLA aligns the VLM’s language predictions with action generation during VLA finetuning.** Standard BC supervises only action prediction, allowing the model’s language output (red) to contradict the required motion on the same observation (blue). Language-Action Alignment derives a discrete motion-direction label from each ground-truth action target and trains the backbone to predict it on the same observation (green), bringing language and action into agreement.

where $\mathbf{W}_{\text{proj}} \in \mathbb{R}^{d \times d}$ is a learned projection and $\mathbf{W}_{\text{lm}} \in \mathbb{R}^{|\mathcal{V}| \times d}$ is the frozen pretrained LM head, so $\hat{\mathbf{a}}^{\text{lang}} \in \mathbb{R}^{|\mathcal{V}|}$ are logits over the vocabulary $\mathcal{V}$. The alignment loss is the cross-entropy of these logits against the ground-truth direction label $\mathbf{a}^{\text{lang}}$:

$$\mathcal{L}_{\text{align}} = \text{CE}(\hat{\mathbf{a}}^{\text{lang}}, \mathbf{a}^{\text{lang}}). \quad (5)$$

Language Label Construction. The alignment targets are derived from ground-truth trajectories with no human annotation. For each training sample, we assign one of six interpretable direction labels, $\mathcal{W} = \{\text{up}, \text{down}, \text{left}, \text{right}, \text{forward}, \text{backward}\}$, through three steps: average chunking, filtering, and discretization. Given the batch of action targets $\mathbf{A} \in \mathbb{R}^{B \times K \times 7}$ ($\mathbf{a}^{\text{cont}}$ stacked over B batch samples, each a K -step chunk), we average the translational components across the chunk dimension to reduce action noise: $\bar{\mathbf{v}} = \frac{1}{K} \sum_{k=1}^K \mathbf{A}_{:,k,1:3} \in \mathbb{R}^{B \times 3}$. We then remove near-stationary samples satisfying $\|\bar{\mathbf{v}}_i\|_2 < \tau$ (where i indexes samples in the batch), with τ a small stationarity threshold. For each remaining sample, we choose the dominant translation axis, $j^* = \arg \max_{j \in \{x,y,z\}} |\bar{v}_{i,j}|$, where $\bar{v}_{i,j}$ is the j -th scalar component of $\bar{\mathbf{v}}_i$, and use the sign of $\bar{v}_{i,j^*}$ to select the corresponding direction word. The resulting direction label $\mathbf{a}^{\text{lang}}$ supervises the language-alignment loss (additional implementation details in App. E.1).

4 Experiments

This section presents our experimental evaluation on simulation benchmarks and real-world experiments. Sec. 4.1 describes the benchmarks, baselines, and implementation details. Sec. 4.2 and Sec. 4.3 present comparisons and ablations on LIBERO-PRO, LIBERO-Plus, and CALVIN simulation benchmarks. Sec. 4.4 shows results on real-world xArm7 manipulation experiments. Finally, Sec. 4.5 provides an in-depth analysis of language preservation and language-action alignment in co-trained VLAs.

4.1 Experimental Setup

Baselines. We compare against baselines covering four families (see App. A for more details): (i) the strong VLA models OpenVLA-OFT [37] and VLA-Adapter [72] (Qwen2.5 [75] backbone, our base model); (ii) the knowledge-preservation baseline VLA-Adapter[Frozen] [72]; (iii) co-trained VLAs MolmoAct [40], ChatVLA [82], Magma [76], and *Co-training + KI*, our reimplement of Knowledge Insulation [21] on VLA-Adapter, which follows the common practice of mixing action finetuning with vision-language supervision [20, 83]: it co-trains on VSR [49], GQA [31], and COCO [47], covering spatial reasoning, compositional VQA, and captioning, and further adopts the KI recipe’s FAST action tokenization [55] and stop-gradient on the VLM backbone; and (iv) the long-horizon CALVIN baselines RoboDual [7], UniVLA [8], MoDE [62], and OpenHelix [12].

Method	LIBERO-PRO				LIBERO-Plus							
	Lang. Reph.	Object Swap	Pos. Swap	Mean	Lang. Instr.	Bg. Text.	Robot Init	Cam. View	Obj. Layout	Light Cond.	Sensor Noise	Mean
Co-training + KI* [21]	54.0	77.4	0.0	43.8	48.0	82.6	25.7	64.6	65.7	73.3	49.0	57.1
MolmoAct [40]	77.8	82.4	0.0	53.4	79.5	84.1	47.4	10.1	76.5	77.4	53.4	60.8
OpenVLA-OFT [37]	74.4	<u>95.2</u>	0.0	56.5	81.5	<u>95.7</u>	40.3	<u>94.7</u>	88.6	<u>95.5</u>	28.2	74.1
VLA-A [Frozen] [72]	56.0	73.4	0.0	43.1	41.5	70.9	35.1	94.4	62.3	84.9	36.2	59.9
VLA-Adapter [72]	<u>91.1</u>	89.6	<u>2.3</u>	<u>61.0</u>	<u>85.1</u>	90.7	<u>52.6</u>	92.6	<u>93.2</u>	93.2	<u>89.5</u>	<u>85.1</u>
Anchor-Align VLA	97.0	96.2	22.6	71.9	87.2	99.6	59.1	96.3	97.4	99.0	96.9	90.3

Table 1: **Success rates on the LIBERO-PRO and LIBERO-Plus benchmarks.** Anchor-Align VLA outperforms all baselines on LIBERO-PRO and LIBERO-Plus. The best result in each column is **bolded** and the second-best is underlined. *Our implementation of knowledge insulation adapted to VLA-Adapter, used to test the efficacy of that approach in the finetuning regime. The 5-seed mean and variance are reported in App. C.3, confirming that these gains are statistically significant.

Benchmarks. We evaluate on LIBERO [48], LIBERO-PRO [80], LIBERO-Plus [22], and CALVIN. LIBERO consists of four 10-task suites (Spatial, Object, Goal, Long) with test setups drawn from the same distribution as the training setup, so it probes only in-distribution performance and is now largely saturated; we nonetheless report standard LIBERO results in the appendix (App. C.1), where Anchor-Align VLA still outperforms all baselines across multiple random seeds. We focus on the more challenging LIBERO-PRO and LIBERO-Plus stress tests below. LIBERO-PRO [80] tests *semantic generalization* via OOD perturbations: language rephrase, object swap, and position swap. LIBERO-Plus [22] tests *perceptual robustness* across seven axes that mirror real-world deployment variability: changes in camera viewpoint, lighting, background texture, object layout, robot initial state, language instruction, and sensor noise. LIBERO-PRO language rephrase rewords the instruction (“pick up the mug” becomes “grab the mug”), whereas LIBERO-Plus language instruction rewrites it globally, e.g., referring to the target by its function (“the flat surface used for holding food” for a plate), forcing the policy to ground the words in the observed scene. Similarly, LIBERO-PRO object swap replaces scene objects with unseen variants and position swap exchanges their locations (Fig. 4), whereas LIBERO-Plus object layout keeps the objects but shifts the target location and adds distractors. Finally, CALVIN ABC→D tests long-horizon, language-conditioned manipulation across four simulated environments (A, B, C, and D): the policy is trained on environments A, B, and C and evaluated zero-shot on the held-out environment D.

Implementation Details. We use Prismatic-Qwen2.5-0.5B [75] as VLM backbone and finetune it with LoRA [29] with rank $r = 64$, applied on all layers. The input sequence is vision + text + proprioceptive state. We use a regression action head with bridge attention to layer-wise condition the action head latents on the VLM features [72]. The 512 vision patches are obtained from two images: we extract DINOv2 [54] and SigLIP [79] features for each image and concatenate them along the feature dimension per patch, yielding 256 patches per image and a 512-patch vision prefix in total. Additional hyperparameters and design choices are discussed in App. B. This is our default architecture. To confirm generality across architectures and action heads, we additionally evaluate on StarVLA [69], an architecturally distinct configuration pairing a Qwen2.5-VL [2] backbone with a flow-matching GR00T FM-DiT [3] action head, in our real-world experiments (Sec. 4.4).

4.2 Simulation Benchmark Comparison

Anchor-Align improves visual grounding and spatial understanding. Tab. 1 shows that our method outperforms all the VLA baselines on every axis of LIBERO-PRO and LIBERO-Plus. For LIBERO-PRO, the position-swap axis is the hardest: task-relevant objects are rearranged, so a policy that memorized fixed scene-to-action mappings fails. Here, all other baselines (MolmoAct, OpenVLA-OFT) score 0% and VLA-Adapter scores 2.3%, while Anchor-Align VLA reaches 22.6%. Fig. 4 shows two qualitative examples from LIBERO-PRO, where Anchor-Align VLA generalizes to the perturbed scene, while standard BC replays its training trajectory.

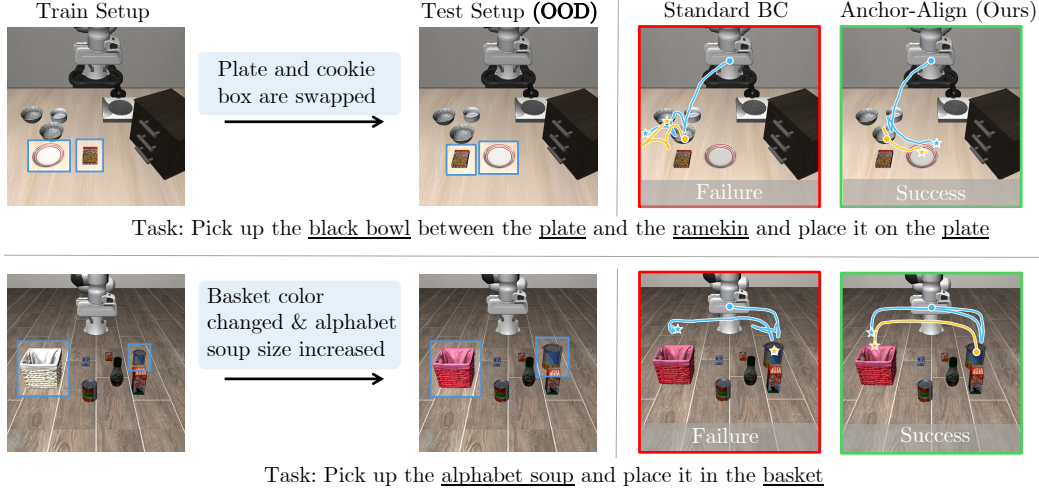


Figure 4: **Anchor-Align VLA generalizes to semantic perturbations.** Both rows are from LIBERO-PRO: the top shows the position-swap axis and the bottom the object-swap axis. Standard BC fails by executing trajectories tied to the training scene, such as reaching toward the original object location or missing the resized target. Anchor-Align VLA grounds to the perturbed observation and succeeds.

Anchor-Align improves robustness to perturbations. LIBERO-Plus consists of challenging perturbations like changing background texture, adding sensor noise, or unseen initial robot arm configurations. As we can see from Tab. 1, Anchor-Align VLA improves over VLA-Adapter [72] on every axis, with the largest gains on the hardest perturbations: background texture (+8.9), sensor noise (+7.4), robot initial state (+6.5), and lighting (+5.8). In absolute terms, Anchor-Align VLA succeeds 99.6% under background texture changes and 99.0% under lighting changes. Per-suite robustness breakdowns for the LIBERO Long, Object, and Goal suites are provided in App. C.2.

Anchor-Align improves long-horizon generalization.

Tab. 2 shows results on the long-horizon CALVIN ABC→D benchmark. In each CALVIN rollout, the policy receives a chain of five language instructions and must complete them consecutively: the $k/5$ columns report the fraction of rollouts completing the first k instructions, and **Len** denotes the average number of consecutively completed tasks (out of five). Because the chain advances only if each instruction succeeds, small grounding errors compound over the rollout. We therefore compare against the baselines most relevant to this claim: OpenVLA-OFT and UniVLA, strong large-scale action-pretrained VLAs; OpenHelix, a recent CALVIN-specific VLA; and VLA-Adapter trained with standard BC (full comparison in App. C.4). Anchor-Align VLA is the best at every chain length and attains the longest average rollout (4.5 vs. 4.3 for VLA-Adapter), with the gain over VLA-Adapter widening on the deepest chains where errors compound, from +0.8 at one instruction to +4.8 at five. It also beats the strongest action-pretrained baseline, OpenVLA-OFT, raising five-instruction completion from 66.5% to 77.9%, indicating that preserving and aligning pretrained VLM representations improves long-horizon generalization without large-scale robot-action pretraining.

CALVIN ABC→D	1/5 (↑)	2/5 (↑)	3/5 (↑)	4/5 (↑)	5/5 (↑)	Len (↑)
UniVLA [8]	95.5	85.8	75.4	66.9	56.5	3.8
OpenVLA-OFT [37]	96.3	89.1	82.4	75.8	66.5	4.1
OpenHelix [12]	97.1	91.4	82.8	72.6	64.1	4.1
VLA-Adapter [72]	98.3	94.0	87.5	80.0	73.1	4.3
Anchor-Align VLA	99.1	95.8	90.6	84.7	77.9	4.5

Table 2: **Long-horizon generalization on CALVIN ABC→D.** **Len** denotes the average rollout length. Full comparison with additional baselines in App. C.4.

4.3 Impact of Anchoring and Alignment

In this section, we present ablations and comparisons to validate our two proposed loss terms, demonstrating that both are needed to get the best performance, and that Language-Action alignment is not just a generic regularization term.

Both anchoring and alignment are necessary. Tab. 3 presents an ablation analysis that isolates the impact of our proposed additional loss terms by comparing the VLA-Adapter baseline with Anchor VLA (only layer-wise MSE distillation loss $\mathcal{L}_{\text{MSE}}$ against a frozen VLM copy), Align VLA (only the language-action alignment loss $\mathcal{L}_{\text{align}}$), and the full Anchor-Align VLA. We can see that each of anchoring and alignment independently improve over standard BC on every axis, and combining them in Anchor-Align VLA exceeds either alone. Thus, the two play complementary roles: anchoring preserves the representational substrate that alignment leverages.

Method	Key Technique	PRO	Plus
VLA-Adapter	<i>Standard BC</i>	61.0	85.1
Align VLA	<i>Alignment only</i>	65.9	88.6
Anchor VLA	<i>Anchoring only</i>	68.1	87.3
Anchor-Align VLA	<i>Full method</i>	71.9	90.3

Table 3: Ablation study on LIBERO-PRO and LIBERO-Plus. PRO and Plus report the mean success rate over the LIBERO-PRO and LIBERO-Plus axes, respectively. Anchor VLA (anchoring only) and Align VLA (alignment only) each improve over baseline.

Alignment is not just regularization. One might wonder whether the gains come from genuine language-action alignment or merely from a regularization effect that any auxiliary classification loss would provide [46]. Is the model improving because each predicted direction word is correctly grounded in its own observation, or simply because adding a six-way word-classification objective regularizes the pre-action state? To separate these two explanations, we train *Shuffle* and *Scatter* controls with the same training setup as Anchor-Align VLA (including anchoring) and the same auxiliary loss, but with corrupted direction labels that break the per-observation language-action alignment while leaving the auxiliary task intact. *Shuffle* applies a fixed permutation to the observation-to-label mapping (e.g., *left*→*forward*, *right*→*up*), so each direction label no longer matches the action on its observation; *Scatter* keeps this mapping but replaces the six direction words with six arbitrary, meaningless ones (e.g., *purple*, *math*), removing their motion semantics. As shown in Tab. 4, a pure regularization effect would survive this corruption; instead, both controls collapse toward standard VLA finetuning, showing that the improvement stems from genuine language-action alignment and not from the mere presence of an auxiliary classification loss (additional details in App. B.5).

Method	Key Technique	PRO	Plus
VLA-Adapter	<i>Standard BC</i>	61.0	85.1
<i>Shuffle</i>	<i>Regularization control</i>	61.4	84.9
<i>Scatter</i>	<i>Regularization control</i>	63.3	85.7
Anchor-Align VLA	<i>Full method</i>	71.9	90.3

Table 4: **Regularization controls.** *Shuffle* and *Scatter* collapse to the VLA-Adapter baseline rather than recovering the gains of Anchor-Align VLA. Mean LIBERO-PRO and LIBERO-Plus scores are reported.

4.4 Real-World Results

In simulation, objects are perfectly rigid and fixed in orientation, perception is clean, and each perturbation changes one aspect of the scene at a time. Next, we test Anchor-Align VLA on a real UFactory xArm7 manipulation setup, where the policy has to deal with varied object orientations, noisy perception and actuation, clutter, and multiple aspects of the scene shifting together.

Our real-world benchmark is designed to test observation-grounded generalization rather than memorization of a single tabletop layout. As shown in Fig. 5, we evaluate four held-out perturbation regimes. **Spatial rearrangement** changes the positions of the target object, surrounding objects, and distractors. **Cluttered scene** (Fig. 6) introduces additional distractor objects and requires the policy to select the object specified by language. **Compositional Object-Layout** jointly changes object placement and object orientation. For both Spatial rearrangement and Compositional Object-Layout, every evaluation is unseen with respect to training data, creating conditions that are more challenging than the position swap of LIBERO-PRO, as here pick, place, and distractor objects are swapped. We further extend Spatial rearrangement into a harder **Semantic perturbation** OOD test that changes the instruction along with the spatial arrangement of these objects: a policy trained to pick up a *green* mug is evaluated with the OOD instruction “pick up a *pink* mug and place it on the plate.” Solving it requires using the preserved VLM color concept under a new instruction rather than defaulting to the training object. Details of hardware, data collection, and deployment for our real-world experiments can be found in App. B.4.

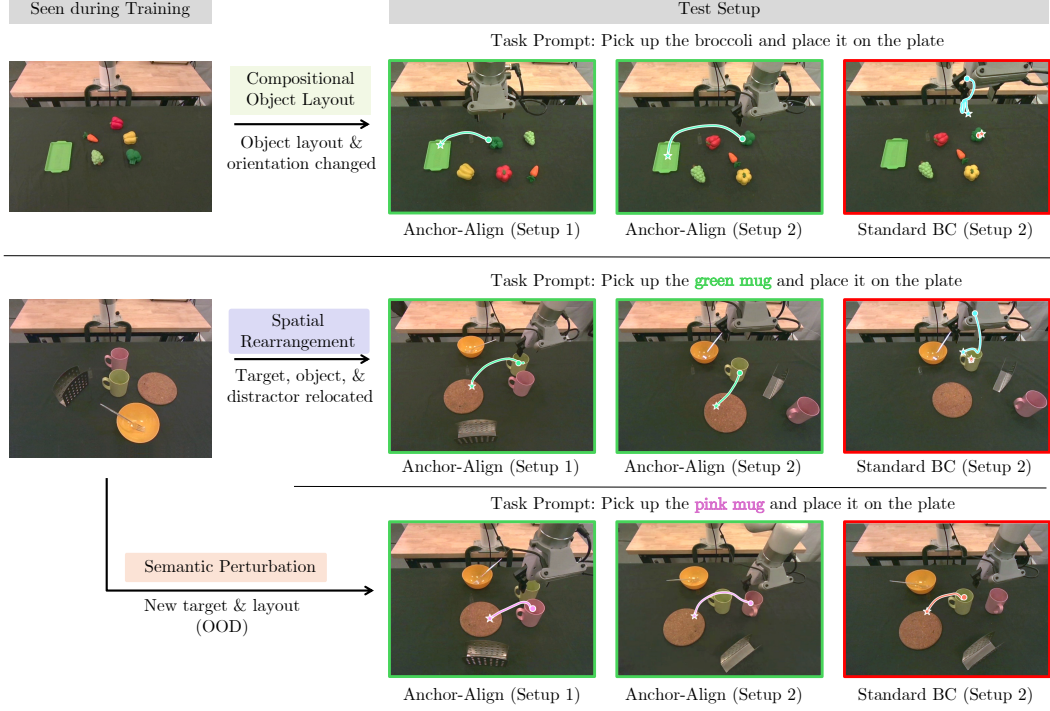


Figure 5: **Real-world rollouts: Anchor-Align generalizes across spatial setups, whereas standard BC fails.** Each row is a held-out perturbation regime (compositional object layout, spatial rearrangement, and semantic perturbation); the layout seen during training (left) always differs from the one at test (right). *Setup 1* and *Setup 2* are two distinct spatial configurations of the same task, with the target, objects, and distractors relocated, probing generalization across spatial distributions rather than memorization of a single layout. Anchor-Align VLA succeeds in both setups (green border), whereas standard BC fails on the same setup (red border). If the object doesn’t move (e.g. Standard BC row 1 & 2), then the gripper trajectory is shown (in blue).

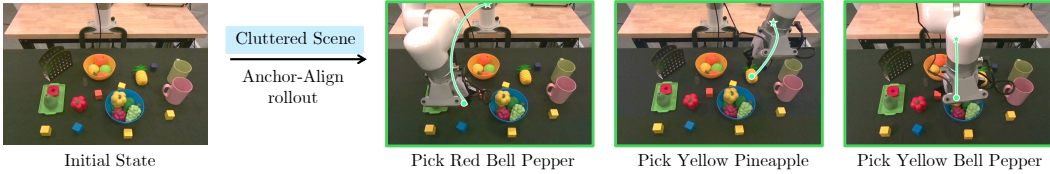


Figure 6: **Cluttered-scene.** Starting from a heavily cluttered tabletop (left), Anchor-Align VLA picks the language-specified target across many distractors, grounding each instruction in the current scene.

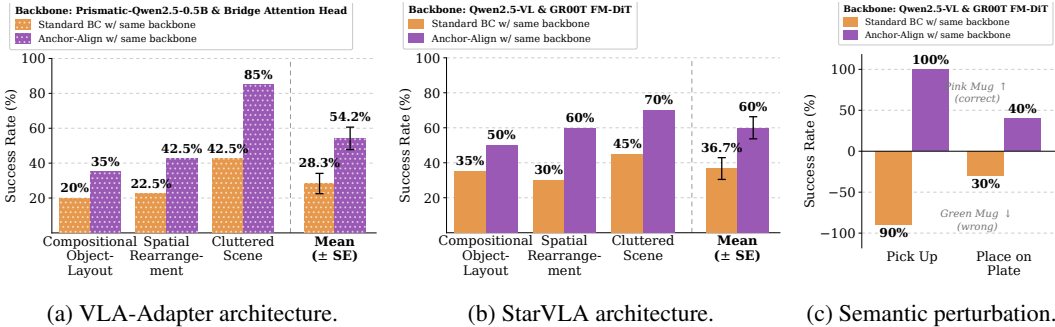


Figure 7: **Real-world generalization across different perturbations and backbones.** (a, b) Success rates on three generalization tests (spatial rearrangement, cluttered scene, and compositional object-layout) for two VLA backbones (20 rollouts per condition). Anchor-Align improves every condition on both backbones, so the gains are not tied to a specific backbone or action head. (c) A pink-mug semantic-perturbation test (trained only on the green mug), where Anchor-Align VLA follows the “pink mug” instruction while the standard BC baseline defaults to its green-mug prior. Here, we evaluate only the StarVLA setup, since the L1 regression head of VLA-Adapter cannot capture multimodal action distributions.

Fig. 7 shows results with two VLA setups: VLA-Adapter [72] with a Prismatic [35]-Qwen2.5-0.5B [75] backbone, and the architecturally distinct StarVLA [69], which pairs Qwen2.5-VL [2] with a GR00T FM-DiT [3] action head. The two setups differ in VLM scale (0.5B vs. 3B), action objective (L1 regression vs. flow matching), and action-head architecture (a bridge-attention regression head vs. a diffusion transformer), respectively. On the hardest test, semantic perturbation, we report results using the stronger StarVLA setup, since the deterministic L1 regression head of VLA-Adapter cannot capture multimodal action distributions [11, 23].

Anchor-Align VLA consistently improves real-world generalization across both architectures (Fig. 7). With the VLA-Adapter setup, mean success rises from 28.3% to 54.2%. With the StarVLA setup, Anchor-Align improves every condition: compositional object-layout from 35% to 50%, spatial rearrangement from 30% to 60%, cluttered scenes from 45% to 70%, and mean success from 36.7% to 60.0%. These gains are therefore not tied to a specific VLM backbone or action head. On the semantic-perturbation test, the standard BC-trained StarVLA baseline collapses to its training prior: it picks the green mug in 90% of trials and places it in 30%. In contrast, Anchor-Align VLA picks the pink mug in 100% of trials and places it on the plate in 40%. This supports our central claim: standard BC binds language to the training scene, while Anchor-Align re-grounds the instruction in the current observation.

Limitations and Error Analysis. We manually inspect all failed real-world rollouts and bin clearly attributable failures into six modes (Fig. 8): *Incorrect Object*, approaches the wrong object; *Semantic Error*, selects a feature-similar distractor (e.g., green grapes instead of green broccoli); *Memorization Error*, ignores the scene and follows the training trajectory; *Grasp Failure*, reaches the target but does not descend to grasp; *Grasp-and-Drop Error*, grasps correctly but releases too early, too high, or not at all; and *Close Miss*, grounds correctly but misses by a small margin. Fig. 8 shows that Anchor-Align VLA eliminates semantic errors ($7 \rightarrow 0$), nearly eliminates incorrect-object approaches ($10 \rightarrow 1$), reduces memorization errors ($15 \rightarrow 9$), and nearly halves grasp failures ($15 \rightarrow 8$). The slight grasp-and-drop increase ($12 \rightarrow 13$) reflects more successful grasps, hence more opportunities for drop errors.

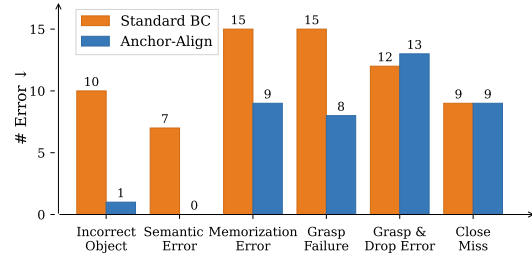


Figure 8: Failure mode analysis comparing standard BC and Anchor-Align.

4.5 Analysis

This section presents an in-depth analysis of language preservation and language-action misalignment in VLA training.

Anchor-Align preserves VLM representations.

For effective generalization, a VLA must be able to perform visual reasoning, i.e., understand objects, their semantics, and their attributes in a scene, an ability that is a hallmark of pretrained VLM representations; GQA [31] tests this ability through compositional questions spanning object, attribute, and spatial understanding. Here, we quantify visual reasoning in standard BC and Anchor-Align, through GQA accuracy over the course of finetuning with the frozen VLM (same VLM used as backbone for VLA model) as an upper bound. As shown in Fig. 9, standard BC degrades monotonically, losing 94% of its GQA accuracy within 10K steps (catastrophic forgetting), whereas Anchor-Align retains 70% of the pretrained accuracy. Correspondingly, LIBERO-PRO task success rates (Table 1) are

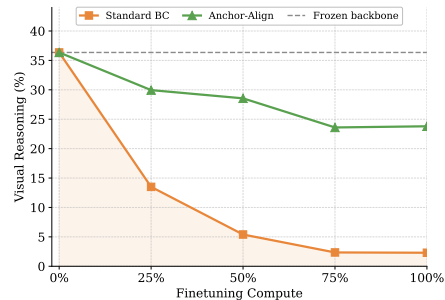


Figure 9: **Visual reasoning accuracy during finetuning.** Standard BC loses 94% of its GQA accuracy within 10K steps (orange), while Anchor-Align retains 70% (green); the frozen VLM is the upper bound (dashed).

71.9% (Anchor-Align), 61.0% (standard BC), and 43.1% (frozen), showing that action performance and vision-language reasoning are not fundamentally at odds. We further substantiate this study by quantifying the change in representation by different finetuning paradigms in App. D.3.

Language-action alignment is positively correlated with task success.

In co-training, the language and action heads are never supervised on the same observation, so nothing forces the model to express the same behavior through both outputs. We measure this misalignment by extending the programmatic framework of Sec. 3.3, which converts robot demonstrations into language labels, into a diagnostic tool: at each timestep, we compare the model’s language prediction with its discretized policy action. The language prediction is obtained by probing the language head with a templated question about the robot’s next motion (exact prompts in App. E), and the continuous policy action is discretized into the same label space, so the two heads are aligned on a frame when they produce the same label. Tab. 5 reports results LIBERO-PRO, we report **success**, the task success rate over the evaluated rollouts, and **alignment**, the fraction of observations on which the two heads agree, regardless of whether that shared prediction is correct. Additionally, we test whether alignment predicts success: we compute the alignment of each individual rollout (the fraction of its frames on which the two heads agree) and report the Pearson r between this per-rollout alignment and the rollout’s success or failure. Here, *Co-training + KI* is our implementation of co-training with Knowledge Insulation [21] on VLA-Adapter; comparing Anchor-Align VLA against it directly tests whether co-training aligns the two heads. As shown in Tab. 5, VLA-Adapter and Co-training + KI are misaligned, and the near-zero Pearson correlation shows that their agreement is uncorrelated with task success. Anchor-Align VLA raises alignment from 16.8% to 78.4% and turns the correlation strongly positive ($-0.03 \rightarrow +0.51$; rollouts with higher alignment are far more likely to succeed), improving task success from 61.0% to 71.9%. We further report this misalignment study on all competitive co-trained VLAs (ChatVLA, MolmoAct, and Magma) in App. F.

VLA Model	Success	Alignment	Pearson r
Co-training + KI [21]	43.8	14.3	+0.10
VLA-Adapter [72]	61.0	16.8	−0.03
Anchor-Align VLA	71.9	78.4	+0.51

Table 5: **Language-action alignment quantification on LIBERO-PRO rollouts.** “Alignment” is the fraction of observations on which the language and action heads agree; “Pearson r ” correlates per-rollout alignment with task success.

5 Conclusion and Future Work

Standard BC overwrites pretrained VLM semantics and co-training leaves VLA language and action misaligned. Anchor-Align fixes both by preserving the VLM’s pretrained representation and jointly supervising language and action with motion-direction labels. It nearly doubles real-robot task success and sharply improves position-swap generalization, showing that effective VLA fine-tuning must preserve semantics while grounding them in action. Crucially, both objectives reuse supervision already contained in the demonstrations, so these gains require no extra data, annotation, or architectural change, and hold across LIBERO-PRO, LIBERO-Plus, CALVIN, and a physical xArm7 under both regression and flow-matching action heads. We thus recast VLA finetuning as preserving and aligning pretrained VLM priors rather than trading them away for control, offering a simple recipe for adapting future vision-language foundation models to embodied tasks.

Finally, the principles of anchoring and alignment may extend beyond VLAs to video world models. For action-conditioned world models adapted from pretrained video generators [32, 45, 63, 65], frozen-teacher regularization could limit representation drift during robot-specific post-training. An analogous alignment objective could enforce action-outcome consistency (e.g., requiring an inverse-dynamics model to recover the conditioning action from the generated transition) and instruction-rollout consistency when language is present. This offers a concrete path toward world models that retain broad spatiotemporal priors while remaining controllable and semantically grounded.

Acknowledgments

We thank R. Yeeshu Dhurandhar, Tina Xu, Sagar Patil, and Daniel Feng for fruitful discussions. This research is based upon work supported by NSF grant CCF-2348624. This work also received support from the School of ICS at UC Irvine and from Modal through cloud compute credits and infrastructure. This research also used the DeltaAI advanced computing and data resource, which is supported by the National Science Foundation under award OAC 2320345 and the State of Illinois. DeltaAI is a joint effort of the University of Illinois Urbana-Champaign and its National Center for Supercomputing Applications. This work was also supported by the U.S. DARPA ECOLE Program under Award No. HR00112390060. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies, either expressed or implied, of DARPA or the U.S. Government. The U.S. Government is authorized to reproduce and distribute reprints for governmental purposes, notwithstanding any copyright annotation therein.

References

- [1] S. Bahl, R. Mendonca, L. Chen, U. Jain, and D. Pathak. Affordances from Human Videos as a Versatile Representation for Robotics. In *Conference on Computer Vision and Pattern Recognition*, 2023.
- [2] S. Bai, K. Chen, X. Liu, J. Wang, W. Ge, S. Song, K. Dang, et al. Qwen2.5-VL Technical Report. *arXiv preprint arXiv:2502.13923*, 2025.
- [3] J. Bjorck, F. Castañeda, N. Cherniadev, X. Da, R. Ding, L. Fan, et al. GR00T N1: An Open Foundation Model for Generalist Humanoid Robots. *arXiv preprint arXiv:2503.14734*, 2025.
- [4] K. Black, N. Brown, D. Driess, A. Esmail, M. Equi, C. Finn, et al. π_0 : A Vision-Language-Action Flow Model for General Robot Control. In *Robotics: Science and Systems*, 2025.
- [5] Y. Bo, A. Soni, S. Srivastava, and M. Khosla. Evaluating representational similarity measures from the lens of functional correspondence. *arXiv preprint arXiv:2411.14633*, 2024.
- [6] A. Brohan, N. Brown, J. Carbajal, Y. Chebotar, J. Dabis, C. Finn, K. Gopalakrishnan, K. Hausman, A. Herzog, J. Hsu, et al. Rt-1: Robotics transformer for real-world control at scale. *arXiv preprint arXiv:2212.06817*, 2022.
- [7] Q. Bu, H. Li, L. Chen, et al. Towards Synergistic, Generalized, and Efficient Dual-System for Robotic Manipulation. *arXiv preprint arXiv:2410.08001*, 2024.
- [8] Q. Bu, Y. Yang, J. Cai, S. Gao, G. Ren, M. Yao, P. Luo, and H. Li. Univla: Learning to act anywhere with task-centric latent actions. *arXiv preprint arXiv:2505.06111*, 2025.
- [9] R. Cadene, S. Alibert, A. Soare, Q. Gallouedec, A. Zouitine, S. Palma, P. Kooijmans, M. Aractingi, M. Shukor, D. Aubakirova, M. Russi, F. Capuano, C. Pascal, J. Choghari, J. Moss, and T. Wolf. LeRobot: State-of-the-art Machine Learning for Real-World Robotics in Pytorch. <https://github.com/huggingface/lerobot>, 2024.
- [10] K. Chen, S. Xie, Z. Ma, P. R. Sanketi, and K. Goldberg. Robo2vlm: Visual question answering from large-scale in-the-wild robot manipulation datasets. *arXiv preprint arXiv:2505.15517*, 2025.
- [11] C. Chi, Z. Xu, S. Feng, E. Cousineau, Y. Du, B. Burchfiel, R. Tedrake, and S. Song. Diffusion policy: Visuomotor policy learning via action diffusion. *Int. J. Robot. Res.*, 2023.
- [12] C. Cui, P. Ding, W. Song, et al. OpenHelix: A Short Survey, Empirical Analysis, and Open-Source Dual-System VLA Model for Robotic Manipulation. *arXiv preprint arXiv:2505.03912*, 2025.

- [13] D. Dalal, U. Mishra, N. Ahuja, and N. Jojic. City Navigation in the Wild: Exploring Emergent Navigation from Web-Scale Knowledge in MLLMs. *arXiv preprint arXiv:2512.15933*, 2025.
- [14] D. Dalal, G. Vashishtha, U. Mishra, J. Kim, M. Kanda, H. Ha, S. Lazebnik, H. Ji, and U. Jain. Constructive Distortion: Improving MLLMs with Attention-Guided Image Warping. *arXiv preprint arXiv:2510.09741*, 2025.
- [15] D. Dalal, M. Kanda, Z. Wang, H. Ji, and U. Jain. Compositional Reasoning via Joint Image and Language Decomposition. In V. Demberg, K. Inui, and L. Marquez, editors, *Findings of the Association for Computational Linguistics: EACL 2026*, pages 5753–5775, Rabat, Morocco, Mar. 2026. Association for Computational Linguistics. ISBN 979-8-89176-386-9. doi:10.18653/v1/2026.findings-eacl.304. URL <https://aclanthology.org/2026.findings-eacl.304/>.
- [16] A. Das, S. Datta, G. Gkioxari, S. Lee, D. Parikh, and D. Batra. Embodied question answering. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1–10, 2018.
- [17] S. Dasari, M. K. Srirama, U. Jain, and A. Gupta. An Unbiased Look at Datasets for Visuo-Motor Pre-Training. In *Conference on Robot Learning*, pages 1183–1198, 2023.
- [18] R. Doshi, H. Walke, O. Mees, S. Dasari, and S. Levine. Scaling cross-embodied learning: One policy for manipulation, navigation, locomotion and aviation. *arXiv preprint arXiv:2408.11812*, 2024.
- [19] A. Douillard, M. Cord, C. Ollion, T. Robert, and E. Valle. PODNet: Pooled Outputs Distillation for Small-Tasks Incremental Learning. In *European Conference on Computer Vision*, 2020.
- [20] D. Driess, F. Xia, M. S. Sajjadi, C. Lynch, A. Chowdhery, B. Ichter, A. Wahid, J. Tompson, Q. Vuong, T. Yu, et al. Palm-e: An embodied multimodal language model. *arXiv preprint arXiv:2303.03378*, 2023.
- [21] D. Driess, J. T. Springenberg, B. Ichter, L. Yu, A. Li-Bell, K. Pertsch, A. Z. Ren, H. Walke, Q. Vuong, L. X. Shi, et al. Knowledge insulating vision-language-action models: Train fast, run fast, generalize better. *arXiv preprint arXiv:2505.23705*, 2025.
- [22] S. Fei, S. Wang, J. Shi, Z. Dai, J. Cai, P. Qian, L. Ji, X. He, S. Zhang, Z. Fei, J. Fu, J. Gong, and X. Qiu. LIBERO-Plus: In-depth Robustness Analysis of Vision-Language-Action Models. *arXiv preprint arXiv:2510.13626*, 2025.
- [23] P. Florence, C. Lynch, A. Zeng, O. A. Ramirez, A. Wahid, L. Downs, A. Wong, J. Lee, I. Mordatch, and J. Tompson. Implicit behavioral cloning. In *Conference on Robot Learning*, pages 158–168. PMLR, 2022.
- [24] Q. Gao, X. Pi, K. Liu, J. Chen, R. Yang, X. Huang, X. Fang, L. Sun, G. Kishore, B. Ai, et al. Do vision-language models have internal world models? towards an atomic evaluation. *arXiv preprint arXiv:2506.21876*, 2025.
- [25] A. Goyal, H. Hadfield, X. Yang, V. Blukis, and F. Ramos. VLA-0: Building State-of-the-Art VLAs with Zero Modification. *arXiv preprint arXiv:2510.13054*, 2025.
- [26] A. Gretton, O. Bousquet, A. Smola, and B. Schölkopf. Measuring Statistical Dependence with Hilbert–Schmidt Norms. In *Int. Conf. Algorithmic Learn. Theory*, pages 63–77. Springer, 2005.
- [27] S. Grover, A. Gopalkrishnan, B. Ai, H. I. Christensen, H. Su, and X. Li. Enhancing Generalization in Vision-Language-Action Models by Preserving Pretrained Representations. *arXiv preprint arXiv:2509.11417*, 2025.

- [28] G. Hinton, O. Vinyals, and J. Dean. Distilling the Knowledge in a Neural Network. *arXiv preprint arXiv:1503.02531*, 2015.
- [29] E. J. Hu, Y. Shen, P. Wallis, Z. Allen-Zhu, Y. Li, S. Wang, L. Wang, and W. Chen. LoRA: Low-Rank Adaptation of Large Language Models. In *ICLR*, 2022.
- [30] C. Huang, M. M. Zhang, R. Azarcon, G. Chou, and Z. Kira. MAPS: Preserving Vision-Language Representations via Module-Wise Proximity Scheduling for Better Vision-Language-Action Generalization. *arXiv preprint arXiv:2511.19878*, 2025.
- [31] D. A. Hudson and C. D. Manning. Gqa: A new dataset for real-world visual reasoning and compositional question answering. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 6700–6709, 2019.
- [32] J. Jang, S. Ye, Z. Lin, J. Xiang, J. Bjoreck, Y. Fang, F. Hu, S. Huang, K. Kundalia, Y.-C. Lin, L. Magne, A. Mandlekar, A. Narayan, Y. L. Tan, G. Wang, J. Wang, Q. Wang, Y. Xu, X. Zeng, K. Zheng, R. Zheng, M.-Y. Liu, L. Zettlemoyer, D. Fox, J. Kautz, S. Reed, Y. Zhu, and L. Fan. DreamGen: Unlocking generalization in robot learning through video world models. *arXiv preprint arXiv:2505.12705*, 2025.
- [33] X. Jiao, Y. Yin, L. Shang, X. Jiang, X. Chen, L. Li, F. Wang, and Q. Liu. TinyBERT: Distilling BERT for Natural Language Understanding. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, 2020.
- [34] N. Kachaev, M. Kolosov, D. Zelezetsky, A. K. Kovalev, and A. I. Panov. Don’t Blind Your VLA: Aligning Visual Representations for OOD Generalization. *arXiv preprint arXiv:2510.25616*, 2025.
- [35] S. Karamcheti, S. Nair, A. Balakrishna, P. Liang, T. Kollar, and D. Sadigh. Prismatic VLMs: Investigating the Design Space of Visually-Conditioned Language Models. In *International Conference on Machine Learning*, 2024.
- [36] M. Kim, K. Pertsch, S. Karamcheti, T. Xiao, A. Balakrishna, S. Nair, R. Rafailov, E. Foster, G. Lam, P. Sanketi, Q. Vuong, T. Kollar, B. Burchfiel, R. Tedrake, D. Sadigh, S. Levine, P. Liang, and C. Finn. OpenVLA: An Open-Source Vision-Language-Action Model. In *Conference on Robot Learning*, 2024.
- [37] M. Kim, C. Finn, and P. Liang. Fine-tuning vision-language-action models: Optimizing speed and success. *arXiv preprint arXiv:2502.19645*, 2025.
- [38] J. Kirkpatrick, R. Pascanu, N. Rabinowitz, J. Veness, G. Desjardins, A. A. Rusu, K. Milan, J. Quan, T. Ramalho, A. Grabska-Barwinska, et al. Overcoming Catastrophic Forgetting in Neural Networks. *Proceedings of the National Academy of Sciences*, 114(13):3521–3526, 2017.
- [39] S. Kornblith, M. Norouzi, H. Lee, and G. Hinton. Similarity of Neural Network Representations Revisited. In *Int. Conf. Mach. Learn.*, pages 3519–3529, 2019.
- [40] J. Lee, J. Duan, H. Fang, Y. Deng, S. Liu, B. Li, B. Fang, J. Zhang, Y. R. Wang, S. Lee, et al. Molmoact: Action reasoning models that can reason in space. *arXiv preprint arXiv:2508.07917*, 2025.
- [41] Z. Li and D. Hoiem. Learning without Forgetting. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 40(12):2935–2947, 2017.
- [42] X. Li, Y. Grandvalet, and F. Davoine. Explicit Inductive Bias for Transfer Learning with Convolutional Networks. In *International Conference on Machine Learning*, 2018.

- [43] X. Li, M. Liu, H. Zhang, C. Yu, J. Xu, H. Wu, C. Cheang, Y. Jing, W. Zhang, H. Liu, et al. Vision-language foundation models as effective robot imitators. *arXiv preprint arXiv:2311.01378*, 2023.
- [44] X. Li, M. Liu, H. Zhang, C. Yu, J. Xu, H. Wu, C. Cheang, Y. Jing, W. Zhang, H. Liu, H. Li, and T. Kong. Vision-Language Foundation Models as Effective Robot Imitators. In *ICLR*, 2024.
- [45] S. Li, Y. Gao, D. Sadigh, and S. Song. Unified Video Action Model. *arXiv preprint arXiv:2503.00200*, 2025.
- [46] L. Liebel and M. Körner. Auxiliary tasks in multi-task learning. *arXiv preprint arXiv:1805.06334*, 2018.
- [47] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick. Microsoft coco: Common objects in context. In *European conference on computer vision*, pages 740–755. Springer, 2014.
- [48] B. Liu, Y. Zhu, C. Gao, Y. Feng, Q. Liu, Y. Zhu, and P. Stone. LIBERO: Benchmarking Knowledge Transfer for Lifelong Robot Learning. In *NeurIPS*, 2023.
- [49] F. Liu, G. Emerson, and N. Collier. Visual Spatial Reasoning. *Transactions of the Association for Computational Linguistics*, 11, 2023.
- [50] A. Majumdar, A. Ajay, X. Zhang, P. Putta, S. Yenamandra, M. Henaff, S. Silwal, P. Mcvay, O. Maksymets, S. Arnaud, et al. Openeqa: Embodied question answering in the era of foundation models. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 16488–16498, 2024.
- [51] J. Mukhoti, Y. Gal, P. H. S. Torr, and P. K. Dokania. Fine-tuning can cripple your foundation model; preserving features may be the solution. *Transactions on Machine Learning Research*, 2024.
- [52] S. Nair, A. Rajeswaran, V. Kumar, C. Finn, and A. Gupta. R3M: A universal visual representation for robot manipulation. In *Conference on Robot Learning*, 2022.
- [53] A. O’Neill, A. Rehman, A. Maddukuri, A. Gupta, A. Padalkar, A. Lee, A. Pooley, A. Gupta, A. Mandlekar, A. Jain, et al. Open x-embodiment: Robotic learning datasets and rt-x models: Open x-embodiment collaboration 0. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pages 6892–6903. IEEE, 2024.
- [54] M. Oquab, T. Darcet, T. Moutakanni, H. Vo, M. Szafraniec, V. Khalidov, P. Fernandez, D. Haziza, F. Massa, A. El-Nouby, et al. Dinov2: Learning robust visual features without supervision. *Transactions on Machine Learning Research*, 2024.
- [55] K. Pertsch, K. Stachowicz, B. Ichter, D. Driess, S. Nair, Q. Vuong, O. Mees, C. Finn, and S. Levine. FAST: Efficient action tokenization for vision-language-action models. *arXiv preprint arXiv:2501.09747*, 2025.
- [56] D. Qu, H. Song, Q. Chen, Z. Chen, X. Gao, X. Ye, Q. Lv, M. Shi, G. Ren, C. Ruan, et al. EO-1: An Open Unified Embodied Foundation Model for General Robot Control. *arXiv preprint arXiv:2508.21112*, 2025.
- [57] D. Qu, H. Song, Q. Chen, Y. Yao, X. Ye, Y. Ding, Z. Wang, J. Gu, B. Zhao, D. Wang, et al. Spatialvla: Exploring spatial representations for visual-language-action model. *arXiv preprint arXiv:2501.15830*, 2025.
- [58] I. Radosavovic, B. Shi, L. Fu, K. Goldberg, T. Darrell, and J. Malik. Robot Learning with Sensorimotor Pre-training. In *Conference on Robot Learning*, pages 683–693, 2023.

- [59] I. Radosavovic, T. Xiao, S. James, P. Abbeel, J. Malik, and T. Darrell. Real-World Robot Learning with Masked Visual Pre-training. In *Conference on Robot Learning*, pages 416–426, 2023.
- [60] S. Ramos, S. Girgin, L. Hussenot, D. Vincent, H. Yakubovich, D. Toyama, A. Gergely, P. Stanczyk, R. Marinier, J. Harmsen, et al. Rlds: an ecosystem to generate, share and use datasets in reinforcement learning. *arXiv preprint arXiv:2111.02767*, 2021.
- [61] S. Reed, K. Zolna, E. Parisotto, S. G. Colmenarejo, A. Novikov, G. Barth-Maron, M. Gimenez, Y. Sulsky, J. Kay, J. T. Springenberg, et al. A generalist agent. *arXiv preprint arXiv:2205.06175*, 2022.
- [62] M. Reuss, J. Pari, P. Agrawal, and R. Lioutikov. Efficient Diffusion Transformer Policies with Mixture of Expert Denoisers for Multitask Learning. In *ICLR*, 2025.
- [63] M. Rigter, T. Gupta, A. Hilmkil, and C. Ma. AVID: Adapting video diffusion models to world models. In *International Conference on Learning Representations*, 2025.
- [64] A. Romero, N. Ballas, S. E. Kahou, A. Chassang, C. Gatta, and Y. Bengio. FitNets: Hints for Thin Deep Nets. In *International Conference on Learning Representations*, 2015.
- [65] S. Routray, H. Pan, U. Jain, S. Bahl, and D. Pathak. ViPRA: Video prediction for robot actions. *arXiv preprint arXiv:2511.07732*, 2025.
- [66] M. Shukor, D. Aubakirova, F. Capuano, P. Kooijmans, S. Palma, A. Zouitine, M. Aractingi, C. Pascal, M. Russi, and A. Marafioti. SmolVLA: A vision-language-action model for affordable and efficient robotics. *arXiv preprint arXiv:2506.01844*, 2025.
- [67] A. Singh, V. Natarajan, M. Shah, Y. Jiang, X. Chen, D. Batra, D. Parikh, and M. Rohrbach. Towards vqa models that can read. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 8317–8326, 2019.
- [68] W. Song, Z. Zhou, H. Zhao, et al. ReconVLA: Reconstructive Vision-Language-Action Model as Effective Robot Perceiver. *arXiv preprint arXiv:2508.10333*, 2025.
- [69] StarVLA Community. StarVLA: A Lego-like Codebase for Vision-Language-Action Model Developing. *arXiv preprint arXiv:2604.05014*, 2026.
- [70] O. M. Team, D. Ghosh, H. Walke, K. Pertsch, K. Black, O. Mees, S. Dasari, J. Hejna, T. Kreiman, C. Xu, et al. Octo: An open-source generalist robot policy. *arXiv preprint arXiv:2405.12213*, 2024.
- [71] L. Wang, X. Chen, J. Zhao, and K. He. Scaling Proprioceptive-Visual Learning with Heterogeneous Pre-trained Transformers. In *Advances in Neural Information Processing Systems*, 2024.
- [72] Y. Wang, P. Ding, L. Li, C. Cui, Z. Ge, X. Tong, W. Song, H. Zhao, W. Zhao, P. Hou, et al. Vla-adapt: An effective paradigm for tiny-scale vision-language-action model. *arXiv preprint arXiv:2509.09372*, 2025.
- [73] J. Wen, Y. Zhu, J. Li, Z. Tang, C. Shen, and F. Feng. Dexvla: Vision-language model with plug-in diffusion expert for general robot control. *arXiv preprint arXiv:2502.05855*, 2025.
- [74] P. Wu, Y. Shentu, Z. Yi, X. Lin, and P. Abbeel. Gello: A general, low-cost, and intuitive teleoperation framework for robot manipulators. In *2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 12156–12163. IEEE, 2024.

- [75] A. Yang, B. Yang, B. Zhang, B. Hui, B. Zheng, B. Yu, C. Li, D. Liu, F. Huang, H. Wei, H. Lin, J. Yang, J. Tu, J. Zhang, J. Yang, J. Yang, J. Zhou, J. Lin, K. Dang, K. Lu, K. Bao, K. Yang, L. Yu, M. Li, M. Xue, P. Zhang, Q. Zhu, R. Men, R. Lin, T. Li, T. Tang, T. Xia, X. Ren, X. Ren, Y. Fan, Y. Su, Y. Zhang, Y. Wan, Y. Liu, Z. Cui, Z. Zhang, Z. Qiu, et al. Qwen2.5 Technical Report. *arXiv preprint arXiv:2412.15115*, 2024.
- [76] J. Yang, R. Tan, Q. Wu, R. Zheng, B. Peng, Y. Liang, Y. Gu, M. Cai, S. Ye, J. Jang, et al. Magma: A foundation model for multimodal ai agents. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pages 14203–14214, 2025.
- [77] Y. Yue, Y. Wang, B. Kang, Y. Han, S. Wang, S. Song, J. Feng, and G. Huang. DeeR-VLA: Dynamic Inference of Multimodal Large Language Models for Efficient Robot Execution. In *NeurIPS*, 2024.
- [78] M. Zawalski, W. Chen, K. Pertsch, O. Mees, C. Finn, and S. Levine. Robotic control via embodied chain-of-thought reasoning. *arXiv preprint arXiv:2407.08693*, 2024.
- [79] X. Zhai, B. Mustafa, A. Kolesnikov, and L. Beyer. Sigmoid loss for language image pre-training. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 11975–11986, 2023.
- [80] X. Zhou, Y. Xu, G. Tie, Y. Chen, G. Zhang, D. Chu, P. Zhou, and L. Sun. LIBERO-PRO: Towards Robust and Fair Evaluation of Vision-Language-Action Models Beyond Memorization. *arXiv preprint arXiv:2510.03827*, 2025.
- [81] Z. Zhou, Y. Zhu, J. Wen, C. Shen, and Y. Xu. ChatVLA-2: Vision-Language-Action Model with Open-World Embodied Reasoning from Pretrained Knowledge. *arXiv preprint arXiv:2505.21906*, 2025.
- [82] Z. Zhou, Y. Zhu, M. Zhu, J. Wen, N. Liu, Z. Xu, W. Meng, R. Cheng, Y. Peng, C. Shen, et al. Chatvla: Unified multimodal understanding and robot control with vision-language-action model. *arXiv preprint arXiv:2502.14420*, 2025.
- [83] B. Zitkovich, T. Yu, S. Xu, P. Xu, T. Xiao, F. Xia, J. Wu, P. Wohlhart, et al. RT-2: Vision-Language-Action Models Transfer Web Knowledge to Robotic Control. In *Proceedings of the 7th Conference on Robot Learning (CoRL)*, volume 229 of *Proceedings of Machine Learning Research*, pages 2165–2183. PMLR, 2023.

Appendix

A: Baseline Details	18
B: Implementation Details	19
B.1 Language-Action Alignment Head	
B.2 Training Setup	
B.3 Training Cost Comparison	
B.4 Real-World Implementation	
B.5 Alignment vs. Regularization: Shuffle and Scatter Controls	
C: Extended Quantitative Results	21
C.1 Success Rates on Standard LIBERO Suites	
C.2 Per-Suite Robustness Breakdowns	
C.3 Multi-Seed Evaluation: Statistical Significance	
C.4 Full CALVIN Comparison	
D: Representation Analysis: Language Preservation and Action Decodability	24
D.1 Centered Kernel Alignment (CKA)	
D.2 Action Decodability (Linear Probing R^2)	
D.3 Language Preservation and Action Decodability	
D.4 Layer-wise Language Preservation	
D.5 Direction Understanding in the Backbone	
E: Language-Action Diagnostic: Dataset Construction	26
E.1 Direction Axis	
E.2 Task Completion Axis	
E.3 Orientation Axis	
E.4 Grasp Axis	
F: Quantification of Misalignment in SOTA Models	29
G: Real-World Rollouts	30
G.1 Faster Task Completion in Real World	
G.2 Object-Orientation Perturbation	
G.3 Anchor-Align Real-World Rollouts	
H: Qualitative Results in LIBERO Simulator	35

A Baseline Details

This appendix extends the condensed baseline list in Sec. 4.1: baselines already introduced there are referenced briefly, and we detail the additional baselines that appear only in the appendix tables.

- **Non-VLM and standard VLM-based VLAs.** OpenVLA-OFT [37] and VLA-Adapter [72] (our base model) are introduced in Sec. 4.1. The standard-LIBERO comparison (Tab. 7) additionally includes Diffusion Policy [11] (non-VLM diffusion-based control), π_0 -FAST [4, 55] (DCT-based action tokenization), SmolVLA [66] (flow-matching action expert), and VLA-0 [25] (plain-text action integers).

- **Co-training baselines.** The co-trained VLAs we compare against are listed in Sec. 4.1; here we summarize what each adds. MolmoAct [40] augments a VLM with depth-aware perception tokens and image-space waypoint planning for 3D spatial reasoning. ChatVLA [81, 82] uses phased alignment training and a mixture-of-experts to retain language capability during action finetuning. Magma [76] adds a Set-of-Mark objective to ground actions within multimodal representations.

- **Knowledge-preservation approaches.** $\pi_{0.5}$ -KI [21] addresses catastrophic forgetting by training discrete FAST action tokens on the VLM backbone while stop-gradient prevents action gradients from corrupting pretrained representations. VLA-Adapter[Frozen] [72] represents the extreme preservation baseline: the VLM backbone is entirely frozen and only the action head is trained, trivially preserving all pretrained knowledge at the cost of reduced task performance. *Co-training + KI*, our reimplementaion of Knowledge Insulation [21] on VLA-Adapter, is already detailed in Sec. 4.1; implementation-wise, action finetuning uses LoRA, the language modeling head is kept

for the VQA forward passes while the L1 regression action head handles robot data, and the FAST-token supervision flows through the language modeling head. This baseline tests whether shielding the VLM from action gradients and co-training on language data is sufficient to prevent catastrophic forgetting.

• **Long-horizon CALVIN baselines.** RoboDual [7], UniVLA [8], MoDE [62], and OpenHelix [12] are introduced in Sec. 4.1. The full CALVIN ABC→D comparison (Tab. 9) additionally includes ReconVLA [68] (auxiliary reconstructive gaze-region grounding for VLA), RoboFlamingo [44] (Open-Flamingo VLM finetuned with an LSTM action head), and DeeR-VLA [77] (dynamic early-exit multimodal inference for efficient robot execution). These methods target long-horizon, instruction-conditioned manipulation under environment shift; none use a representation-preservation or language-action alignment objective on the VLM backbone.

B Implementation Details

For better reproducibility of our results, we include the full architectural specification, hyperparameter settings, and training cost breakdown below.

B.1 Language-Action Alignment Head

The alignment objective (Sec. 3.3) introduces a single learned linear projection $\mathbf{W}_{\text{proj}} \in \mathbb{R}^{d \times d}$ with $d = 896$ (the hidden dimension of Qwen2.5-0.5B), adding $896^2 + 896 = 803,712$ parameters (1.5 MB in bfloat16). At each training step, we take the last-layer hidden state $\mathbf{h}_i^{\text{pre}} = \mathbf{H}^S[N] \in \mathbb{R}^d$ of the last of the N vision and text tokens, the position immediately before the first action token (located in code by scanning the label tensor for the first non-masked entry, which marks the start of the action tokens). It is then mapped through $\mathbf{W}_{\text{proj}}$ and the frozen pretrained language modeling head $\mathbf{W}_{\text{lm}} \in \mathbb{R}^{|\mathcal{V}| \times d}$ to produce the language logits $\hat{\mathbf{a}}_i^{\text{lang}} \in \mathbb{R}^{|\mathcal{V}|}$ over the full vocabulary ($|\mathcal{V}| = 151,646$). A cross-entropy loss is then computed between $\hat{\mathbf{a}}_i^{\text{lang}}$ and the ground-truth direction label $\mathbf{a}_i^{\text{lang}}$ (one of *forward*, *backward*, *left*, *right*, *up*, *down*), with near-stationary samples ($\|\tilde{\mathbf{v}}_i\|_2 < \tau$, $\tau = 0.15$) masked via `ignore_index`. Because $\mathbf{W}_{\text{lm}}$ is frozen, the alignment gradient flows exclusively through $\mathbf{W}_{\text{proj}}$ and the backbone’s LoRA parameters, encouraging the pre-action representation to stay readable by the frozen language head. The projection head is the only additional trainable module beyond the LoRA adapters and action head already present in VLA-Adapter.

B.2 Training Setup

All models are trained on 4 NVIDIA GH200 GPUs using bfloat16 mixed precision with PyTorch 2.7.

Hyperparameters. To ensure reproducibility, we report the complete training configuration used for the results in Tab. 1. The backbone VLM receives two camera images (third-person and wrist) along with the proprioceptive state as input. We use LoRA-based parameter-efficient finetuning with rank $r=64$ and scaling factor $\alpha=128$, applied to all linear layers in the backbone, and merge the LoRA weights into the base model periodically during training. Training runs for 10,000 gradient steps with a batch size of 32, a learning rate of 2×10^{-4} , and standard image augmentation (random crops and color jitter). Checkpoints are saved every 2,500 steps. For the anchoring objective, we apply layer-wise MSE distillation ($\mathcal{L}_{\text{MSE}}$) across all 24 transformer layers with anchoring weight $\lambda_{\text{anchor}}=0.1$. For the alignment objective ($\mathcal{L}_{\text{align}}$), we use an alignment weight of $\lambda_{\text{align}}=0.02$ and a motion-magnitude threshold $\tau=0.15$ to filter near-stationary samples. All remaining hyperparameters, including optimizer settings and data preprocessing, follow the recommended defaults from [72] for the baseline setup.

B.3 Training Cost Comparison

A key practical question is how much overhead knowledge-preservation methods add to standard action finetuning. On identical hardware, we compare Anchor-Align against **Standard BC** (VLA-Adapter [72]) and **Co-training + KI** [21], both detailed in Sec. 4.1. Co-training is the natural point of comparison because it explicitly targets the same goal as our method: retaining the VLM’s pretrained visuolinguistic priors during finetuning. Tab. 6 summarizes the cost and performance trade-offs.

	Standard BC	Co-training + KI*	Anchor-Align
Wall-clock (s/it)	1.28	2.50	1.64
Overhead vs. baseline	—	+95%	+28%
2,500 steps (min)	53	104	68
Extra GPU memory	—	+5 GB	+0.7 GB
External data required	None	25K VQA	None
LIBERO-Plus Overall	85.1	57.1	90.3

Table 6: Training cost and downstream performance comparison. Anchor-Align preserves the VLM’s pre-trained visuolinguistic priors far more effectively than co-training while requiring $3.4\times$ less overhead per step, no external data, and $7\times$ less additional GPU memory. *Our implementation of [21] on VLA-Adapter.

Co-training + KI adds a full VQA forward and backward pass (with batch size 16) at every training step, incurring +95% wall-clock overhead and +5 GB of GPU memory for the additional activations and gradients. It further requires curating 25K external VQA samples (VSR [49], GQA [31], COCO [47]) and maintaining the language modeling head alongside the action head during training. Despite this cost, the resulting policy substantially underperforms: 54.0% on LIBERO-PRO language rephrase and 57.1% LIBERO-Plus overall, both *far below* standard BC (91.1% and 85.1%), indicating that shielding the backbone from action gradients limits its adaptation to the downstream task.

In contrast, Anchor-Align adds only a single inference-only forward pass through a frozen copy of the pretrained VLM (the *anchor*). We replace the anchor’s language modeling head ($151,646 \times 896 \approx 136\text{M}$ parameters) with `nn.Identity()`, since only intermediate hidden states are needed for the layer-wise MSE loss, reducing the anchor’s footprint to 0.7 GB. This yields +28% wall-clock overhead (0.36 s per step), requires no external data or annotation, and completes a full 2,500-step run in 68 minutes—36 minutes faster than co-training. Critically, Anchor-Align achieves 97.0% on language rephrase and 90.3% on LIBERO-Plus overall, demonstrating that preserving the VLM’s representational geometry via anchoring is both cheaper and dramatically more effective than co-training on external VQA data. For larger backbone VLMs, the relative overhead would decrease further, since the anchor’s inference-only forward pass scales sublinearly compared to the training backward pass.

B.4 Real-World Implementation

The real-world setup uses a UFactory xArm7 equipped with a wrist-mounted RGB camera and an external front-view camera, evaluated on tabletop tasks requiring object discrimination, spatial reasoning, and instruction-conditioned target selection under clutter. We collect 150 demonstration trajectories using a 3D-printed GELLO teleoperation device [74] with velocity control at 30 Hz; observation and action spaces are joint angles, and trajectories are stored in the LeRobot [9] format and converted to RLDS [60] for our training pipeline. Both baseline and Anchor-Align VLA are trained for 15K steps with identical hyperparameters, and we report the best-performing checkpoint. Evaluation covers 7 held-out tasks with 20 rollouts per task (140 total trials). At deployment, the low-level velocity controller runs at 100 Hz while the learned policy predicts actions at 30 Hz; commands are interpolated and velocity-clipped for safe execution.

Method	Spatial	Object	Goal	Long
Diffusion Policy [11]	78.3	92.5	68.3	50.5
π_0 -FAST [4, 55]	87.0	63.0	89.0	48.0
SmolVLA-0.24B [66]	87.0	93.0	88.0	63.0
SmolVLA-2.25B [66]	93.0	94.0	91.0	77.0
OpenVLA-OFT [37]	94.3	95.2	91.7	86.5
MolmoAct [40]	87.0	95.4	87.6	77.2
$\pi_{0.5}$ -KI [21]	96.6	97.2	94.6	85.8
VLA-0 [25]	93.6	96.0	95.6	87.6
VLA-Adapter[Frozen] [72]	89.4	89.6	88.0	84.5
VLA-Adapter [72]	96.0	99.8	96.0	89.0
Anchor-Align VLA	98.4	100.0	97.2	90.8

Table 7: **Success rates across standard LIBERO suites.** Anchoring and alignment achieve state-of-the-art results on unperturbed (standard) benchmarks. Best per column is **bolded**. Multi-seed (5 seeds) results for LIBERO-Spatial confirming statistical significance are reported in App. C.3.

B.5 Alignment vs. Regularization: Shuffle and Scatter Controls

This section gives the full construction of the *Shuffle* and *Scatter* controls; their motivation and headline results are covered in Sec. 4.3 and Tab. 4. Both controls are trained with hyperparameters identical to Anchor-Align VLA: the only quantity that changes is the lookup table from motion class to target token id, so each control preserves the loss surface of $\mathcal{L}_{\text{align}}$ while destroying its semantic content.

Shuffle. The observation-to-label mapping is reshuffled once and held fixed throughout training, so the loss magnitude, gradient norm, and update frequency match the genuine alignment loss step-for-step. Concretely, we apply a fixed derangement of the six direction words: *forward* $\rightarrow$ *right*, *backward* $\rightarrow$ *down*, *left* $\rightarrow$ *forward*, *right* $\rightarrow$ *up*, *up* $\rightarrow$ *left*, and *down* $\rightarrow$ *backward*.

Scatter. Each direction class is mapped one-to-one to a fixed meaningless word: *forward* $\rightarrow$ *purple*, *backward* $\rightarrow$ *seven*, *left* $\rightarrow$ *today*, *right* $\rightarrow$ *music*, *up* $\rightarrow$ *tree*, and *down* $\rightarrow$ *math*. The six words are chosen to be mutually distant in the frozen output-embedding space, and the targets remain perfectly synchronized with the observations, so the model still solves a well-defined six-way classification task with an identical loss structure; only the motion semantics of the target tokens are removed.

Per axis, the collapse reported in Tab. 4 is sharpest on the LIBERO-PRO position-swap axis, where success requires re-grounding language to the perturbed scene rather than memorizing scene-to-action mappings: Anchor-Align VLA achieves 22.6% on position swap while *Shuffle* and *Scatter* recover only the baseline’s near-zero performance.

C Extended Quantitative Results

This section complements the main paper’s quantitative evaluation (Sec. 4.2) with success rates on the standard LIBERO suites, per-suite robustness breakdowns, multi-seed significance tests, and the full CALVIN comparison.

C.1 Success Rates on Standard LIBERO Suites

We report success rates on the four standard (unperturbed) LIBERO suites—Spatial, Object, Goal, and Long—in Tab. 7. Anchor-Align VLA achieves the highest success rate on most suites, surpassing prior methods including $\pi_{0.5}$ -KI and OpenVLA-OFT, which use substantially larger backbones and large-scale robotic pretraining. This confirms that the gains from anchoring and alignment hold not only on the OOD stress tests reported in the main paper (Tab. 1), but also on the in-distribution benchmark.

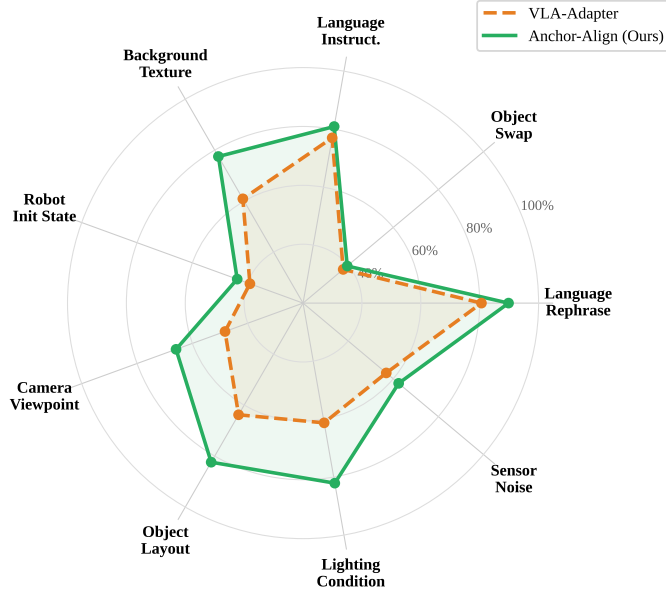


Figure 10: **LIBERO Long Suite**. Performance across nine evaluation axes (excluding Standard and Position Swap). Anchor-Align VLA (orange) substantially expands coverage over VLA-Adapter (gray), with the largest gains on Lighting Condition (+20.8%), Object Layout (+18.6%), and Camera Viewpoint (+17.7%).

C.2 Per-Suite Robustness Breakdowns

In the main paper (Tab. 1), we report robustness and generalization results on the LIBERO-Spatial suite, where perturbations to object positions, language instructions, and scene layout directly test whether the policy has learned transferable spatial reasoning or merely memorized training trajectories. Here we show that the same gains carry over to the remaining three LIBERO suites: Long (Fig. 10), Object (Fig. 11), and Goal (Fig. 12), which test long-horizon composition, object knowledge, and procedural reasoning, respectively. All comparisons are against VLA-Adapter, the strongest standard BC baseline from Tab. 1. Because VLA-Adapter uses the same LoRA architecture and training data as Anchor-Align VLA but without anchoring or alignment, it directly isolates the effect of our two proposed objectives. Each radar plot reports performance across nine evaluation axes: two from LIBERO-PRO (Language Rephrase and Object Swap) and seven from LIBERO-Plus (Language Instruction, Background Texture, Robot Init State, Camera Viewpoint, Object Layout, Lighting Condition, and Sensor Noise). Standard and Position Swap axes are excluded.

C.3 Multi-Seed Evaluation: Statistical Significance

To confirm that the gains reported in the main paper’s Tab. 7 (LIBERO-Spatial) and Tab. 1 (LIBERO-PRO and LIBERO-Plus) are not artifacts of a single random seed, we re-run both Anchor-Align VLA and the VLA-Adapter baseline with five independent training seeds and report mean $\pm$ standard deviation in Tab. 8. The standard deviations are small (all ≤ 0.7) and the mean improvements of Anchor-Align VLA over VLA-Adapter on every column are substantially larger than the combined seed-to-seed variability, so the differences are statistically significant by a paired comparison. The multi-seed means also closely match the single-seed numbers reported in Tabs. 1 and 7, indicating that the headline results are stable rather than seed-lucky.

Gains transfer to a larger VLM backbone. To verify that the improvements from Anchor-Align are not specific to our default backbone or action head, we re-instantiate the method on a different StarVLA configuration than the one used in the main-paper real-world experiments (Sec. 4.4). Recall that StarVLA is a modular VLA framework that combines an arbitrary VLM backbone with an arbitrary action head; here we instantiate it with a larger Qwen2.5 3B VLM and an OFT MLP-based action head, in contrast to the Qwen2.5-VL + GR00T FM-DiT instantiation used in the main paper.

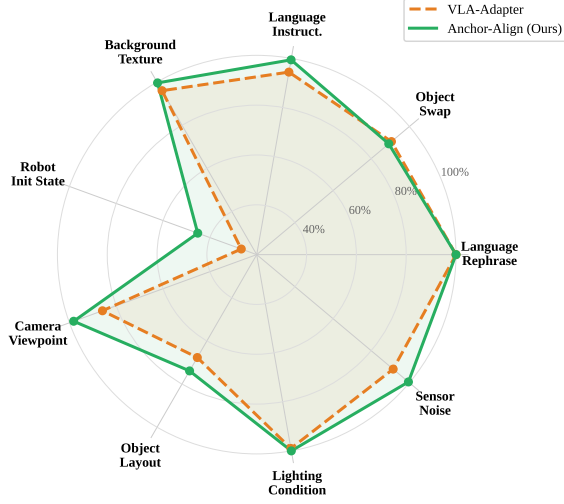


Figure 11: **LIBERO Object Suite**. Performance across nine evaluation axes (excluding Standard and Position Swap). Anchor-Align VLA (orange) consistently covers a larger area than VLA-Adapter (gray), with the most pronounced gains on Robot Init State (+18.6%).

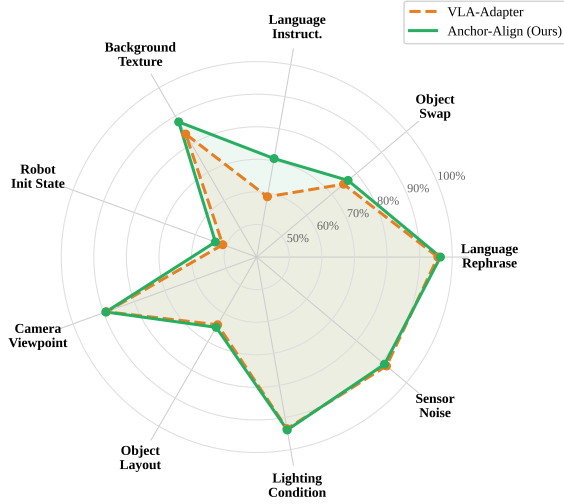


Figure 12: **LIBERO Goal Suite**. Performance across nine evaluation axes (excluding Standard and Position Swap). Anchor-Align VLA (orange) shows the largest gains over VLA-Adapter (gray) on Language Instruction (+11.9%) and Background Texture (+4.3%).

On LIBERO-PRO, Anchor-Align VLA improves the StarVLA baseline from 73.6% to **89.0%** on the language rephrase axis and from 89.8% to **91.2%** on the object swap axis. The improvement is qualitatively consistent with what we observe on the smaller backbone, indicating that the benefits of anchoring and alignment are a property of the training objective rather than of a particular VLM scale or action-head design.

C.4 Full CALVIN Comparison

The main paper’s Tab. 2 reports a compact CALVIN ABC→D comparison against the most relevant baselines; Tab. 9 lists the full set. Anchor-Align VLA outperforms all baselines at every chain length and attains the longest average rollout.

Method	Spatial	Lang. Reph.	Object Swap	Pos. Swap	Plus
VLA-Adapter	93.3 $\pm$ 0.3	91.1 $\pm$ 0.4	90.1 $\pm$ 0.5	2.6 $\pm$ 0.7	85.3 $\pm$ 0.3
Anchor-Align VLA	97.9 $\pm$ 0.3	97.1 $\pm$ 0.5	96.1 $\pm$ 0.4	23.5 $\pm$ 0.2	90.5 $\pm$ 0.6

Table 8: **Multi-seed evaluation (5 seeds)**. Mean $\pm$ standard deviation of success rate (%) for Anchor-Align VLA and the VLA-Adapter baseline on LIBERO-Spatial (standard), LIBERO-PRO language rephrase / object swap / position swap, and the LIBERO-Plus overall average. Seed-to-seed variability is small (≤ 0.7) and substantially smaller than the gap between methods, confirming that the improvements reported in the main paper’s Tabs. 1 and 7 are statistically significant.

CALVIN ABC $\rightarrow$ D	1/5 ($\uparrow$)	2/5 ($\uparrow$)	3/5 ($\uparrow$)	4/5 ($\uparrow$)	5/5 ($\uparrow$)	Len ($\uparrow$)
RoboFlamingo [44]	82.4	61.9	46.6	33.1	23.5	2.5
DeeR-VLA [77]	86.2	70.1	51.8	41.5	30.4	2.8
RoboDual [7]	94.4	82.7	72.1	62.4	54.4	3.7
UniVLA [8]	95.5	85.8	75.4	66.9	56.5	3.8
ReconVLA [68]	95.6	87.6	76.9	69.3	64.1	4.0
MoDE [62]	96.2	88.9	81.1	71.8	63.5	4.0
OpenVLA-OFT [37]	96.3	89.1	82.4	75.8	66.5	4.1
OpenHelix [12]	97.1	91.4	82.8	72.6	64.1	4.1
VLA-Adapter [72]	<u>98.3</u>	<u>94.0</u>	<u>87.5</u>	<u>80.0</u>	<u>73.1</u>	<u>4.3</u>
Anchor-Align VLA	99.1	95.8	90.6	84.7	77.9	4.5

Table 9: **Full CALVIN ABC $\rightarrow$ D comparison**. Anchor-Align VLA outperforms all listed CALVIN baselines at every horizon. **Len** denotes the average rollout length. The best result in each column is **bolded** and the second-best is underlined.

D Representation Analysis: Language Preservation and Action Decodability

This appendix provides the full mathematical details for the evaluation metrics used in the paper, followed by the representation-level analyses built on them.

D.1 Centered Kernel Alignment (CKA)

We use CKA [5, 39] to measure how much of the pretrained VLM’s representational geometry is preserved after finetuning. CKA is built on the Hilbert–Schmidt Independence Criterion (HSIC) [26], which quantifies the statistical dependence between two sets of representations via their kernel matrices.

HSIC. Given n data points and two representation matrices $\mathbf{X} \in \mathbb{R}^{n \times p}$ and $\mathbf{Y} \in \mathbb{R}^{n \times q}$, we form the Gram matrices $\mathbf{K} = \mathbf{X}\mathbf{X}^\top$ and $\mathbf{L} = \mathbf{Y}\mathbf{Y}^\top$ (under a linear kernel). The empirical HSIC is

$$\text{HSIC}(\mathbf{K}, \mathbf{L}) = \frac{1}{(n-1)^2} \text{tr}(\mathbf{K} \mathbf{H} \mathbf{L} \mathbf{H}), \quad (6)$$

where $\mathbf{H} = \mathbf{I}_n - \frac{1}{n} \mathbf{1}\mathbf{1}^\top$ is the centering matrix. Centering removes the mean from each kernel matrix, so that HSIC captures the structural (second-order) similarity rather than mere magnitude.

CKA. CKA normalizes HSIC so that the score lies in $[0, 1]$:

$$\text{CKA}(\mathbf{K}, \mathbf{L}) = \frac{\text{HSIC}(\mathbf{K}, \mathbf{L})}{\sqrt{\text{HSIC}(\mathbf{K}, \mathbf{K}) \text{HSIC}(\mathbf{L}, \mathbf{L})}}. \quad (7)$$

A CKA score of 1.0 indicates that the two representations have identical geometric structure up to an isotropic scaling and orthogonal transformation; lower values indicate that finetuning has reshaped the layer’s representations.

Linear CKA. When the linear kernel is used, the CKA expression simplifies to a closed-form that avoids explicitly constructing the $n \times n$ Gram matrices. Let $\tilde{\mathbf{X}}$ and $\tilde{\mathbf{Y}}$ denote the column-centered

versions of $\mathbf{X}$ and $\mathbf{Y}$. Then:

$$\text{CKA}_{\text{linear}}(\mathbf{X}, \mathbf{Y}) = \frac{\|\tilde{\mathbf{Y}}^\top \tilde{\mathbf{X}}\|_F^2}{\|\tilde{\mathbf{X}}^\top \tilde{\mathbf{X}}\|_F \cdot \|\tilde{\mathbf{Y}}^\top \tilde{\mathbf{Y}}\|_F}. \quad (8)$$

This is the form we use throughout: at each transformer layer u , $\mathbf{X} \in \mathbb{R}^{n \times d}$ contains the text-token hidden states from the frozen pretrained VLM and $\mathbf{Y} \in \mathbb{R}^{n \times d}$ from the finetuned backbone VLM, across n evaluation samples. We report text-token CKA because it directly reflects the degree to which the model’s language understanding is preserved.

D.2 Action Decodability (Linear Probing R^2)

To measure how much action-relevant information is linearly accessible from the model’s hidden states, we train a ridge regression probe at each layer. For layer u , we extract the hidden state and mean-pool over all token positions to obtain a feature vector $\mathbf{f}_i^{(u)} \in \mathbb{R}^d$ for each sample i . We then fit a ridge regression model g to predict the ground-truth discretized action $\mathbf{a}_i \in \mathbb{R}^7$ (with 256 bins per dimension):

$$R^2(u) = 1 - \frac{\sum_{i=1}^n \|\mathbf{a}_i - g(\mathbf{f}_i^{(u)})\|_2^2}{\sum_{i=1}^n \|\mathbf{a}_i - \bar{\mathbf{a}}\|_2^2}, \quad (9)$$

where $\bar{\mathbf{a}}$ is the mean action over the dataset. A higher R^2 indicates that more action information is linearly decodable from that layer’s representations, even without the action head.

D.3 Language Preservation and Action Decodability

We quantify what each training paradigm does to the backbone’s internal representations along two axes: language preservation, measured with Centered Kernel Alignment (CKA) [5] between the finetuned backbone’s text-token hidden states and the pretrained VLM’s (formal definition in App. D.1), and action decodability, measured by linear-probing the backbone’s hidden states for the ground-truth action (details in App. D.2). As shown in Fig. 13, standard BC catastrophically destroys the pretrained text representations (CKA drops to 0.34; per-layer analysis in App. D.4). At the opposite extreme, the frozen backbone trivially preserves the pretrained geometry yet gains no action information, and the co-trained VLA behaves similarly. Anchor-Align VLA achieves the best of both: it maintains strong language preservation (CKA = 0.91) while attaining the highest action decodability of any method (peak $R^2 = 0.60$ at layer 22), showing that alignment routes action-relevant information through the decoder layers without overwriting the pretrained language geometry. Preservation captures what the model keeps from pretraining, decodability what it gains from finetuning; together they show that Anchor-Align VLA attains high task performance by enriching the pretrained representations rather than destroying them, as standard BC does. Additional linear-probe results on direction concepts (in-distribution vs. OOD) are reported in App. D.5.

D.4 Layer-wise Language Preservation

App. D.3 reports language preservation per model; here we break the same measurement down by decoder layer, asking whether the GQA collapse in Fig. 9 is only behavioral or whether finetuning actually reshapes the backbone’s internal representations. Fig. 14 plots the text-token CKA between the finetuned backbone and the pretrained VLM at every layer (formal definition in App. D.1). Standard BC progressively destroys the pretrained text representations with depth, collapsing in the output layers (CKA drops to 0.34 at layer 24), the same layers that drive the GQA collapse, so the degradation is representational rather than merely behavioral. Anchor VLA instead recovers near-perfect preservation through layer-wise distillation, with an average CKA of 0.95 across layers.

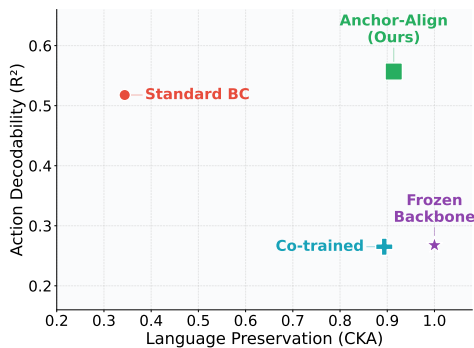


Figure 13: **Action decodability vs. language preservation.** Per-model language preservation (text-token CKA) vs. action decodability (linear-probe R^2): Anchor-Align VLA attains the highest action decodability while sustaining high language preservation, gaining action information without overwriting the pretrained geometry.

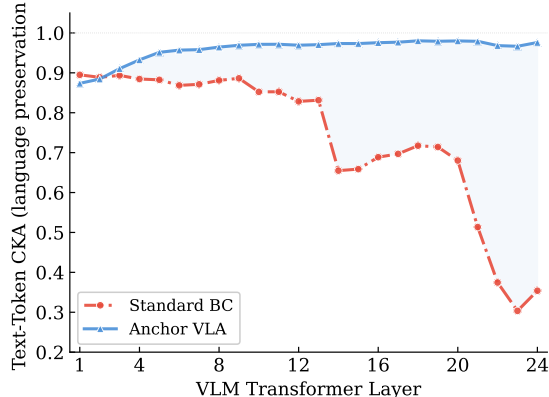


Figure 14: **Layer-wise language preservation.** Per-layer text-token CKA between the finetuned backbone and the pretrained VLM. Standard BC collapses the pretrained text representations in the output layers, whereas Anchor VLA preserves them across all layers through layer-wise distillation.

D.5 Direction Understanding in the Backbone

To further unpack what Anchor-Align changes inside the backbone, we linearly probe motion-direction concepts from the backbone’s hidden states on spatial (in-distribution) and object (out-of-distribution) splits. Tab. 10 shows that Anchor-Align VLA substantially improves directional decodability, with the largest relative improvement on the OOD object split. Combined with the end-to-end language-action alignment results reported in Tab. 11, this indicates that the anchoring objective preserves a backbone in which directional concepts remain linearly accessible, and that the alignment objective then converts this latent structure into coherent joint language-action behavior at inference time.

Method	Spatial (in-dist.)	Object (OOD)
VLA-Adapter	86.0	21.4
Anchor-Align VLA	94.4	30.3

Table 10: **Direction understanding.** Linear-probe accuracy on motion-direction concepts shows that Anchor-Align VLA preserves directional structure inside the backbone, with the largest relative improvement on the OOD object split. End-to-end Direction-axis alignment metrics for VLA-Adapter and Anchor-Align VLA are reported in Tab. 11.

E Language-Action Diagnostic: Dataset Construction

This section details the programmatic curation pipeline for the language-action diagnostic introduced in Sec. 4.5. The diagnostic is built by a general-purpose programmatic framework that extracts language labels from robot trajectory data. The main paper applies it along a single axis, *motion direction*, the axis we follow throughout the paper; here we show that the same framework extends to three additional axes: end-effector *orientation* change, *grasp* state, and *task completion* status (Fig. 15). Together, the four axes cover the key functional capabilities required for manipulation; each probes a distinct aspect of the VLA’s internal understanding (coarse spatial planning, fine rotational control, contact reasoning, and progress monitoring) and is evaluated independently, so that deficiencies along any single axis can be isolated.

We use this extended diagnostic to quantify misalignment in state-of-the-art co-trained VLAs (App. F). Most of these models do not release LIBERO checkpoints, so the extended diagnostic

cannot be run in the LIBERO simulator; instead, we evaluate on the real robot images such pretrained models are trained for. We therefore apply the framework to the MolmoAct [40] mid-training dataset, which spans 73 distinct manipulation tasks across diverse household environments, object categories, and interaction types. The dataset consists of episodes recorded with 7-DoF action vectors (3 translational, 3 rotational, 1 gripper); this task diversity ensures that the extracted labels are not biased toward any single manipulation skill or scene configuration. Moreover, because we evaluate pretrained checkpoints on recorded frames rather than deploying them as policies, the diagnostic reports per-frame (frame-level) language, action, and alignment accuracies rather than rollout task success. Below we describe the ground-truth label generation procedure for each axis.

The **direction** split contains 10,000 frames classified into six direction words (left, right, up, down, forward, backward); the distribution is naturally skewed toward downward approach movements (32.1%), with backward motion being the rarest (5.1%). The **orientation** split uses 1,001 frames uniformly distributed across 7 classes (no-rotation, roll-cw/ccw, pitch-up/down, and yaw-left/right; 143 per class), achieved by balanced sampling from the raw trajectory pool. **Task completion** comprises 10,000 frames with a binary label (yes/no) split as 33.6%/66.4%. The **grasp** split contains 5,000 frames labeled as open or closed ($\sim 2:1$ open-to-closed).

E.1 Direction Axis

The direction axis evaluates whether the VLA’s language and action heads agree on the translational motion direction; it is the axis used in the main paper (Secs. 3.3 and 4.5). The ground-truth labels are generated by the programmatic procedure introduced in Sec. 3.3, which maps chunked action vectors to one of six direction words via average chunking, filtering, and discretization. This subsection adds the complete per-frame formulation, with which we label the MolmoAct dataset used in App. F to expose the misalignment of pretrained co-trained VLAs.

Given a robot trajectory of T frames, each frame t is associated with an action vector $\mathbf{a}_t = (\Delta x_t, \Delta y_t, \Delta z_t, \dots) \in \mathbb{R}^7$, where the first three components represent translational end-effector displacements in world coordinates. We extract the translational component $\mathbf{d}_t = (\Delta x_t, \Delta y_t, \Delta z_t)$ and assign a single direction label via dominant-axis discretization.

Dominant-axis assignment. We identify the axis with the largest absolute displacement:

$$j^* = \arg \max_{j \in \{x, y, z\}} |\mathbf{d}_{t,j}| \quad (10)$$

with tie-breaking priority $x > y > z$. The sign of $\mathbf{d}_{t,j^*}$ then selects one of six direction words ($\Delta x \mapsto$ forward/backward, $\Delta y \mapsto$ left/right, $\Delta z \mapsto$ up/down), yielding a single label ℓ_t^* per frame, consistent with the discretization in Sec. 3.3.

Filtering. Two filters are applied before a frame is retained in the training set:

1. *Magnitude filter:* $\|\mathbf{d}_t\|_2 \geq \mu$ with $\mu = 0.001$, removing near-stationary frames.
2. *Consistency filter:* the top-1 label ℓ_t^* must equal ℓ_{t+j}^* for all $j \in \{1, \dots, K\}$ with $K = 5$, ensuring temporally stable motion intent and suppressing noisy transient frames.

A frame is retained only if both conditions are satisfied.

E.2 Task Completion Axis

The 7-DoF action vector at frame t includes a gripper command $g_t = \mathbf{a}_t[6] \in [0, 1]$, where values below a threshold $\theta_g = 0.5$ indicate an open gripper and values above indicate a closed gripper. We define a binary openness indicator: $o_t = 1$ if $g_t < \theta_g$ (open) and $o_t = 0$ otherwise (closed). We then identify the task-completion boundary by detecting the last gripper state transition in the episode. For example, in a pick-and-place task the final transition corresponds to the object being released at the target location. Formally, we define $t_{\text{end}} = \max\{t : o_{t-1} \neq o_t\}$ as the frame of the last gripper state change. Episodes with no detectable transition are excluded. Each frame receives

a binary label c_t : frames before the last transition ($t < t_{\text{end}}$) are labeled $c_t = 0$ (incomplete), while frames at or after it ($t \geq t_{\text{end}}$) are labeled $c_t = 1$ (complete).

Training formulation. Each sampled frame is paired with a task-conditioned binary question: “*In this current state do you think the robot has completed the task: $\{task\}$? Answer in one word yes or no.*” The ground-truth answer is “yes” if $c_t = 1$ and “no” otherwise.

E.3 Orientation Axis

Orientation labels are derived from the rotational components of the 7-DoF action vector $\mathbf{a}_t = (\cdot, \cdot, \cdot, \Delta\phi_t, \Delta\theta_t, \Delta\psi_t, \cdot)$, where $\Delta\phi$, $\Delta\theta$, $\Delta\psi$ denote instantaneous roll, pitch, and yaw deltas, respectively. Unlike the per-frame labeling used for translational direction (Sec. 3.3), orientation labels are computed over a sliding window of $W = 7$ consecutive frames to accumulate sufficient rotational signal. For a window starting at frame t , the cumulative orientation change is:

$$(\Phi, \Theta, \Psi) = \left(\sum_{k=t}^{t+W-1} \Delta\phi_k, \sum_{k=t}^{t+W-1} \Delta\theta_k, \sum_{k=t}^{t+W-1} \Delta\psi_k \right) \quad (11)$$

with cumulative magnitude $M = \sqrt{\Phi^2 + \Theta^2 + \Psi^2}$.

Two-stage filtering. Two thresholds handle sensor noise and near-stationary orientations:

1. *Noise gate* ($\mu_{\text{noise}} \approx 0.57^\circ$): windows with $M < \mu_{\text{noise}}$ are discarded entirely as sensor noise.
2. *No-rotation threshold* ($\mu_{\text{rot}} \approx 1^\circ$): windows with $\mu_{\text{noise}} \leq M < \mu_{\text{rot}}$ are labeled as `no_rotation`.

Dominant-axis classification. For windows with $M \geq \mu_{\text{rot}}$, the label is determined by the axis with the largest absolute cumulative change and its sign:

$$k^* = \arg \max_{k \in \{0,1,2\}} |[\Phi, \Theta, \Psi]_k| \quad (12)$$

This yields six directional labels. Together with the `no_rotation` class from the filtering stage, the full 7-class label space is $\mathcal{Y}_{\text{orient}} = \{\text{no_rotation}, \text{roll_cw}, \text{roll_ccw}, \text{pitch_up}, \text{pitch_down}, \text{yaw_left}, \text{yaw_right}\}$.

Dataset construction. The sliding window is applied densely across each episode with a stride of 1 frame (not W), yielding one sample per valid starting frame. Each sample stores the first frame’s index (used for image retrieval during evaluation), the list of frame indices in the window, the cumulative (Φ, Θ, Ψ) values, the magnitude M , and the derived label.

E.4 Grasp Axis

Grasp labels are derived from the gripper command $g_t = \mathbf{a}_t[6] \in [0, 1]$, the same scalar used in the task-completion labeling (Sec. E.2). Whereas task completion uses the gripper timeline to locate the final state transition, the grasp category directly classifies the instantaneous gripper state into a binary label $\ell_t^g \in \{\text{open}, \text{closed}\}$ via the threshold $\theta_g = 0.5$.

Episode selection. Only episodes that exhibit a complete pick-and-place gripper pattern are retained. We define a *pick* event as the first frame t_p where the gripper transitions from open to closed ($o_{t_p-1} = 1$, $o_{t_p} = 0$, where $o_t = 1$ if $g_t < \theta_g$), and a *place* event as the first subsequent frame t_r where the gripper reopens ($o_{t_r-1} = 0$, $o_{t_r} = 1$). Episodes lacking either event are excluded.

Stage partitioning and sampling. Each qualifying episode is partitioned into three functional stages based on t_p and t_r :

1. *Pre-grasp* (frames $[0, t_p-1]$): the end-effector approaches the object with an open gripper.
2. *Grasp-to-release* (frames $[t_p, t_r-1]$): the object is grasped and transported.
3. *Post-release* (frames $[t_r, T]$): the gripper reopens after placing the object.

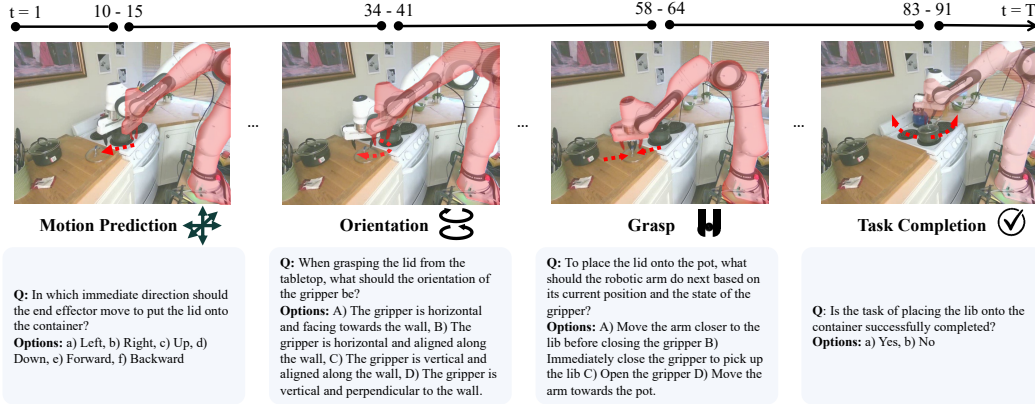


Figure 15: **Illustrating all four diagnostic axes.** Each robot manipulation episode is segmented into four diagnostic axes: Motion Direction ($t=1$ to 15), Orientation ($t=34$ to 41), Grasp ($t=58$ to 64), and Task Completion ($t=83$ to 91). For each axis, ground-truth labels are extracted and paired with template questions for evaluation.

$N_s = 10$ frames are sampled uniformly at random from each stage via seeded sampling, yielding up to $3N_s = 30$ labeled frames per episode. This three-stage design captures both gripper states across all manipulation phases, though the resulting label distribution is naturally imbalanced ($\sim 2:1$ open-to-closed) because stages 1 and 3 predominantly contain open-gripper frames.

F Quantification of Misalignment in SOTA Models

Using the diagnostic dataset constructed in App. E, this section quantifies language-action misalignment in state-of-the-art co-trained VLAs (ChatVLA, MolmoAct, and Magma), extending the analysis of Sec. 4.5 from the motion-direction axis to all four diagnostic axes. As explained in App. E, these models are evaluated per frame on real robot images, so we report frame-level Language, Action, and Alignment accuracies rather than rollout task success.

Evaluation distribution for Tab. 11. We evaluate the three co-trained VLAs (ChatVLA, MolmoAct, and Magma) on the MolmoAct mid-training dataset [40], chosen because it consists of real robot images: these pretrained models are built for real images, and evaluating them in simulation would place them out of distribution.

VLA Model	Direction $\uparrow$			Task Completion $\uparrow$			Grasp $\uparrow$			Orientation $\uparrow$		
	Align.	Lang.	Action	Align.	Lang.	Action	Align.	Lang.	Action	Align.	Lang.	Action
ChatVLA	20.9	20.3	20.4	48.1	65.6	44.3	47.7	50.4	66.0	20.7	12.9	14.1
MolmoAct	15.5	22.6	41.3	24.6	44.6	65.2	37.8	51.8	77.1	7.4	15.0	28.1
Magma	17.2	27.3	28.0	39.0	56.6	65.8	40.8	49.4	49.5	12.6	13.6	13.9

Table 11: **Analysis of language-action alignment across VLAs using our diagnostic.** We report Language and Action accuracy (%) against ground truth, and Alignment (%), the fraction of frames on which the two heads agree, across the Direction, Task Completion, Grasp, and Orientation axes. Prior co-trained VLAs are poorly aligned on every axis.

Tab. 11 reports all four axes of our diagnostic. The Task Completion axis tests whether the VLA can recognize terminal states (object placed, gripper released, etc.); Grasp tests contact-event detection; Orientation tests mid-trajectory reorientation. Across the three additional axes, the qualitative pattern matches Direction: even when the language and action heads each achieve moderate accuracy, their joint alignment is substantially lower, with MolmoAct showing the largest per-head-vs.-joint gap on Task Completion (44.6% language and 65.2% action, yet only 24.6% alignment). Below we analyze the structure of this misalignment along each axis.

Direction and Orientation. These fine-grained spatial axes exhibit the lowest alignment scores across all models (7.4–20.9%), reflecting the fact that translational and rotational planning are processed through largely decoupled pathways. MolmoAct’s action head reaches 41.3% on direction, well above the 16.7% chance level for 6-way classification, while its language head scores only 22.6%. Yet the two agree on just 15.5% of timesteps. This shows that the action head has learned directional control that the language channel cannot describe. This misalignment is precisely the gap that our alignment objective targets: by training the pre-action hidden state to predict direction words derived from ground-truth actions, we force the backbone to route directional information through a shared representation that is accessible to both heads.

Task Completion. All three models achieve alignment scores of 24.6–48.1% on this axis, but the gap between per-head accuracy and joint accuracy is striking. ChatVLA predicts task completion correctly 65.6% of the time from language and 44.3% of the time from action, yet the two heads agree only 48.1% of the time. This means that a substantial fraction of correct language predictions co-occur with incorrect action predictions, and vice versa. Importantly, the alignment score also counts cases in which both heads make the same incorrect prediction; even under this lenient criterion, the near-random alignment score clearly indicates misalignment. This suggests that each head learns a partially correct but independent signal for terminal-state detection, rather than sharing a unified internal representation of task progress.

Grasp. The grasp results are consistent with the previous finding of misalignment between heads. MolmoAct’s action head achieves 77.1% on grasp detection, while its language head reaches only 51.8%; yet alignment is only 37.8%. This means that the action head has learned reliable grasp control that the language channel cannot describe, and that the two heads frequently disagree on the same observation. Conversely, Magma’s language and action heads both hover near 49–50% (near-random performance), while alignment is 40.8%, again near chance. This indicates that even when the individual head accuracies are similar, the two heads are not coordinating their predictions; instead, they arrive at similar aggregate statistics through different per-timestep decisions.

G Real-World Rollouts

G.1 Faster Task Completion in Real World

Beyond improving success rates, Anchor-Align also produces notably faster task execution in the real world. Fig. 16 compares the distribution of rollout durations across successful trials for standard BC (VLA-Adapter) and Anchor-Align VLA on the xArm7 pick-and-place task. Anchor-Align VLA completes rollouts $1.7\times$ faster on average, with a tighter distribution of completion times.

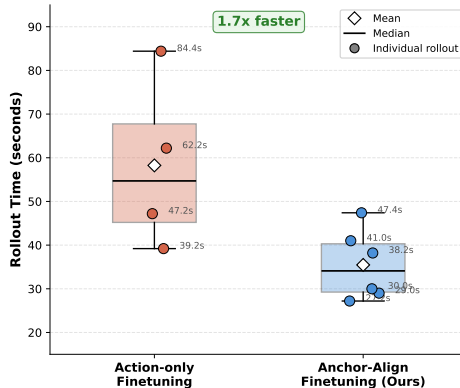


Figure 16: **Real-world rollout completion time.** Distribution of successful rollout durations on the xArm7 pick-and-place task. Anchor-Align VLA completes rollouts $1.7\times$ faster than standard BC, with lower variance.

We attribute this speedup to qualitatively different action vectors produced by the two methods. Near the grasp point—the most critical phase of the trajectory—Standard BC outputs smaller, more tentative action vectors. These low-magnitude predictions frequently undershoot the required displacement, leading to repeated corrective adjustments before the gripper achieves a secure grasp. In contrast, Anchor-Align VLA produces higher-magnitude, more decisive action vectors at the grasp point, achieving a firm and consistent grasp on the first attempt. We hypothesize that the alignment objective, by grounding the backbone’s representations in interpretable language labels (e.g., “down” during the approach phase), encourages the model to commit to coherent motion plans rather than hedge with small incremental steps.

This decisiveness propagates beyond the grasp: once the object is secured, Anchor-Align VLA executes the transport and placement phases with equally direct trajectories. The result is not only higher success rates but also more efficient, more natural-looking robot behavior—an important practical consideration for real-world deployment where cycle time directly affects throughput.

G.2 Object-Orientation Perturbation

Beyond swapping object positions, our Compositional Object-Layout regime also perturbs the *orientation* of the target object itself. Fig. 17 shows the broccoli used in the pick-and-place task in six distinct poses: varying the object’s orientation while holding its identity and the instruction fixed prevents the policy from matching a memorized canonical view and forces it to re-ground perception in each rollout.

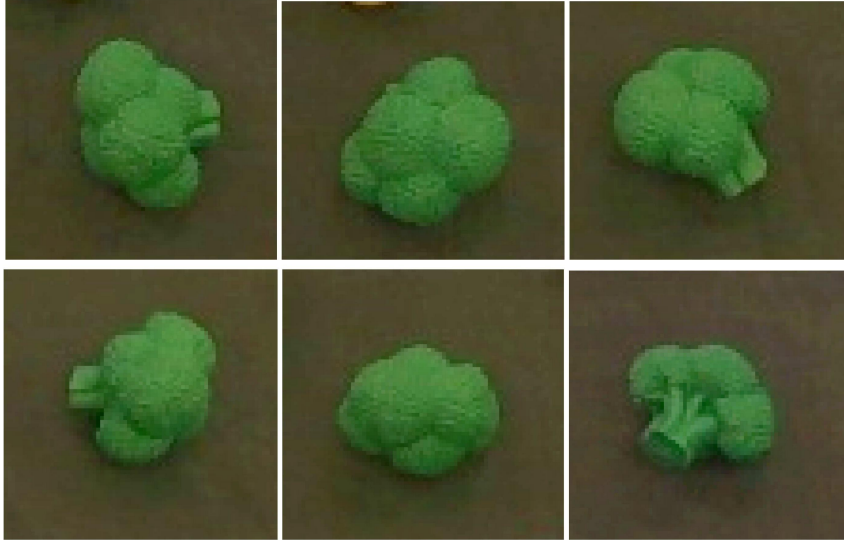


Figure 17: **Target-object orientation perturbation (Compositional Object-Layout).** The same broccoli, cropped from the front-view camera, shown in six distinct orientations used across evaluation rollouts of the “Pick up the green broccoli and place it on the plate task”. We perturb the orientation of the object *itself*, not just its position, so its silhouette and visible structure change substantially while its identity and the language instruction remain fixed. This prevents the policy from relying on a canonical, memorized view of the target and, combined with the position and distractor swaps in Fig. 18, makes the task considerably harder than the position-only swap of LIBERO-PRO.

G.3 Anchor-Align Real-World Rollouts

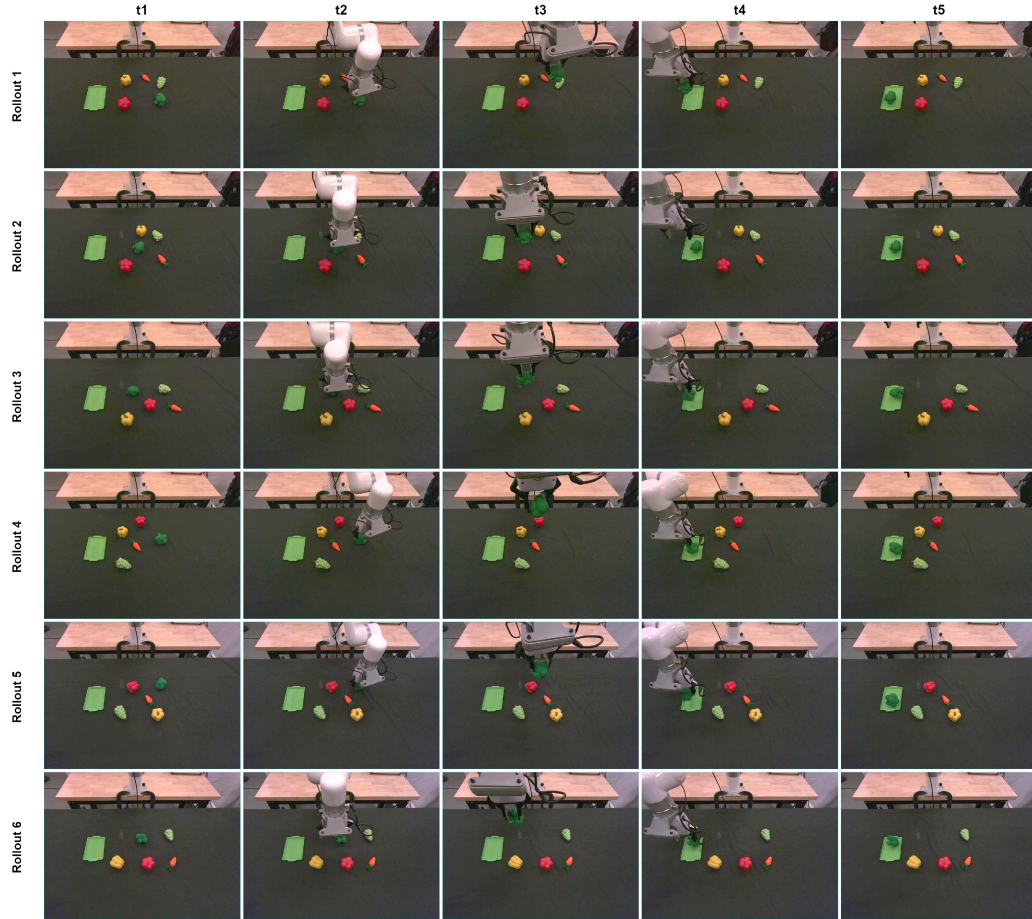


Figure 18: **Successful real-world rollouts from the Anchor-Align VLA model.** The task is to pick up the broccoli and place it on the green plate. **The broccoli’s spatial position and the surrounding distractor objects are simultaneously swapped across rollouts**, so each rollout is evaluated on a unique scene configuration. All six rollouts succeed, showing that Anchor-Align VLA handles joint object-swap and position-swap perturbations robustly without relying on memorized scene configurations. Each row shows 5 evenly spaced keyframes from a single rollout.

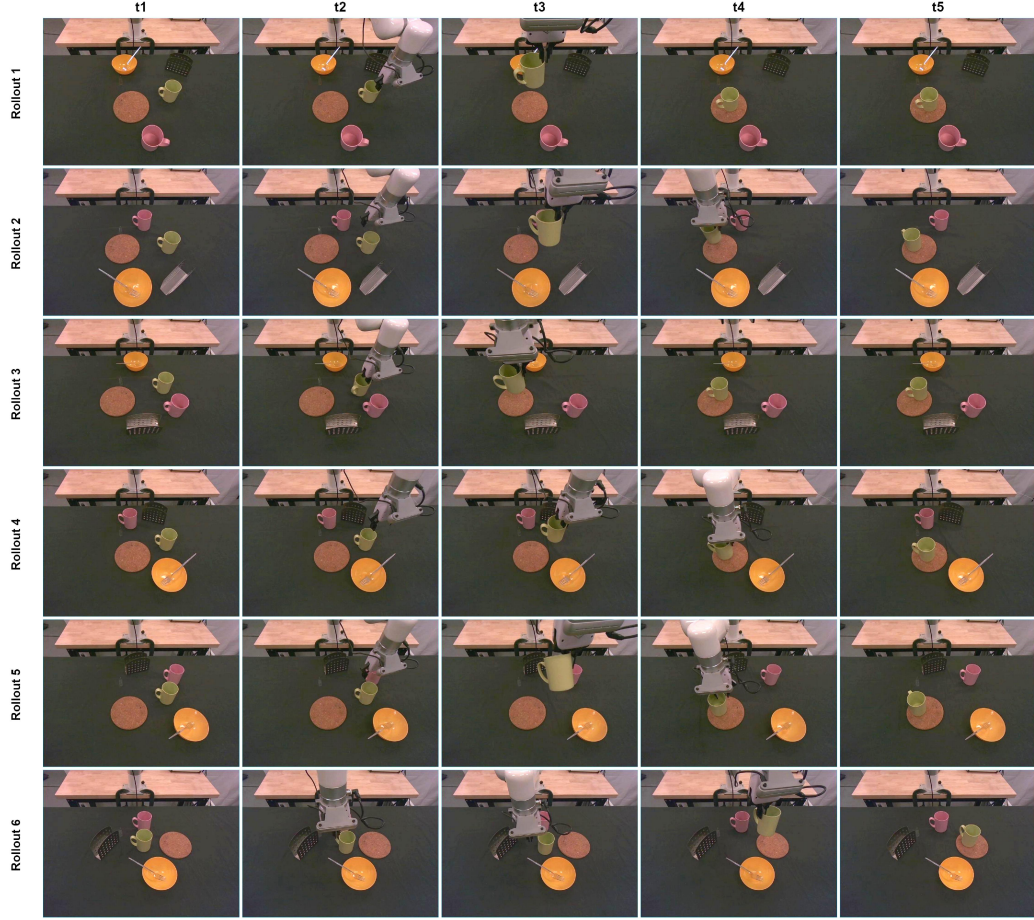


Figure 19: **Successful real-world rollouts from the Anchor-Align VLA model.** The task is to pick up the green mug and place it on the plate. **The mug’s spatial position and the surrounding distractor objects are simultaneously swapped across rollouts**, so each rollout is evaluated on a unique scene configuration. All six rollouts succeed, showing that Anchor-Align VLA handles joint object-swap and position-swap perturbations robustly without relying on memorized scene configurations. Each row shows 5 evenly spaced keyframes from a single rollout.



Figure 20: **Successful real-world rollouts from the Anchor-Align VLA model.** The task is to pick up the object named by the language instruction (pineapple, red bell pepper, or yellow bell pepper) and place it on the plate in a **very cluttered scene**. **The target is specified by text while many distractor objects (including the other candidate items) crowd the workspace**, so the policy must ground the instruction in the current observation and grasp the correct object rather than defaulting to a memorized choice. All six rollouts succeed, showing that Anchor-Align VLA discriminates the language-specified target under heavy clutter and visually similar distractors. Each row shows 5 evenly spaced keyframes from a single rollout.

H Qualitative Results in LIBERO Simulator

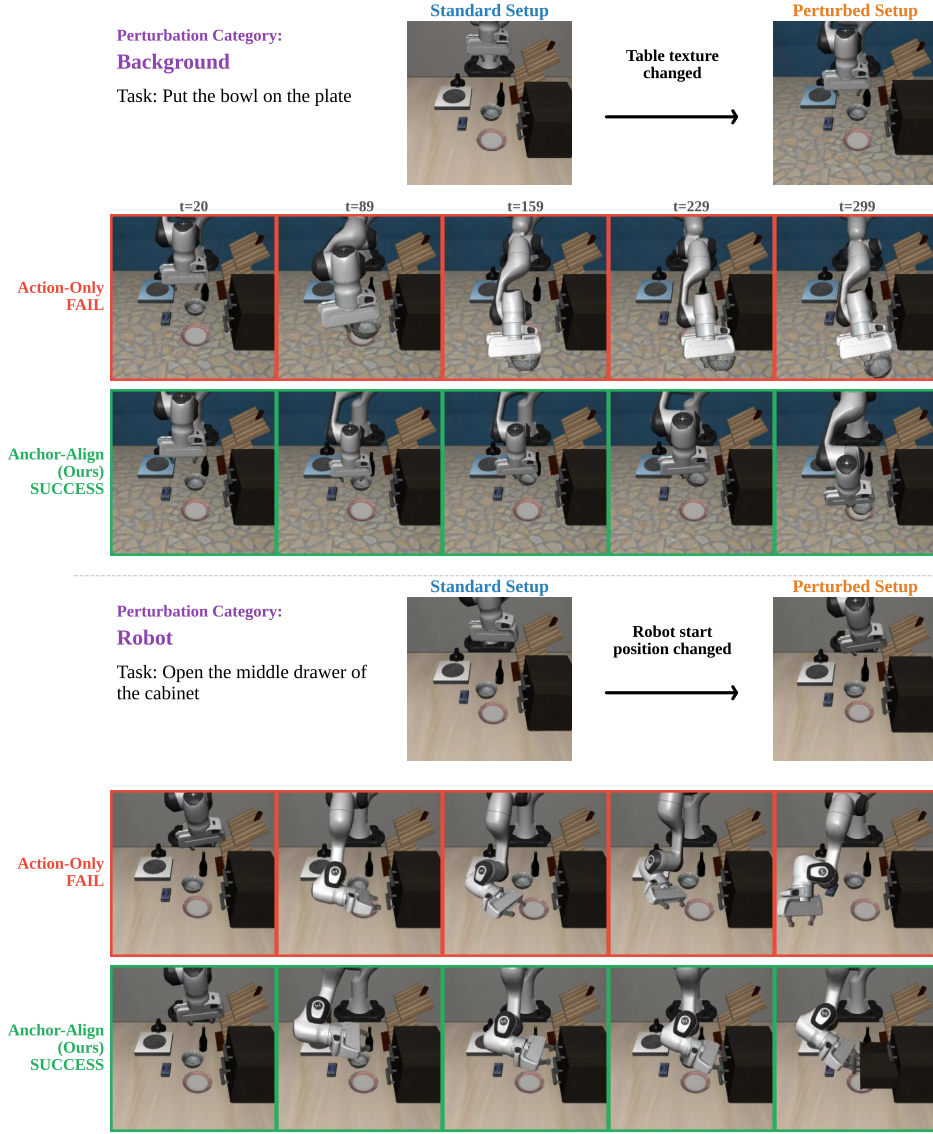


Figure 21: **LIBERO-Goal Plus: Background and Robot perturbations.** *Top:* Background perturbation on “Put the bowl on the plate”—the table texture is changed to stone tile. Standard BC (red) collapses onto the table without releasing the bowl on the plate; Anchor-Align VLA (green) places the bowl on the plate. *Bottom:* Robot perturbation on “Open the middle drawer of the cabinet”—the robot’s initial joint configuration is shifted. Standard BC approaches the wrong side of the cabinet and never opens the drawer; Anchor-Align VLA locates the middle drawer and pulls it open.

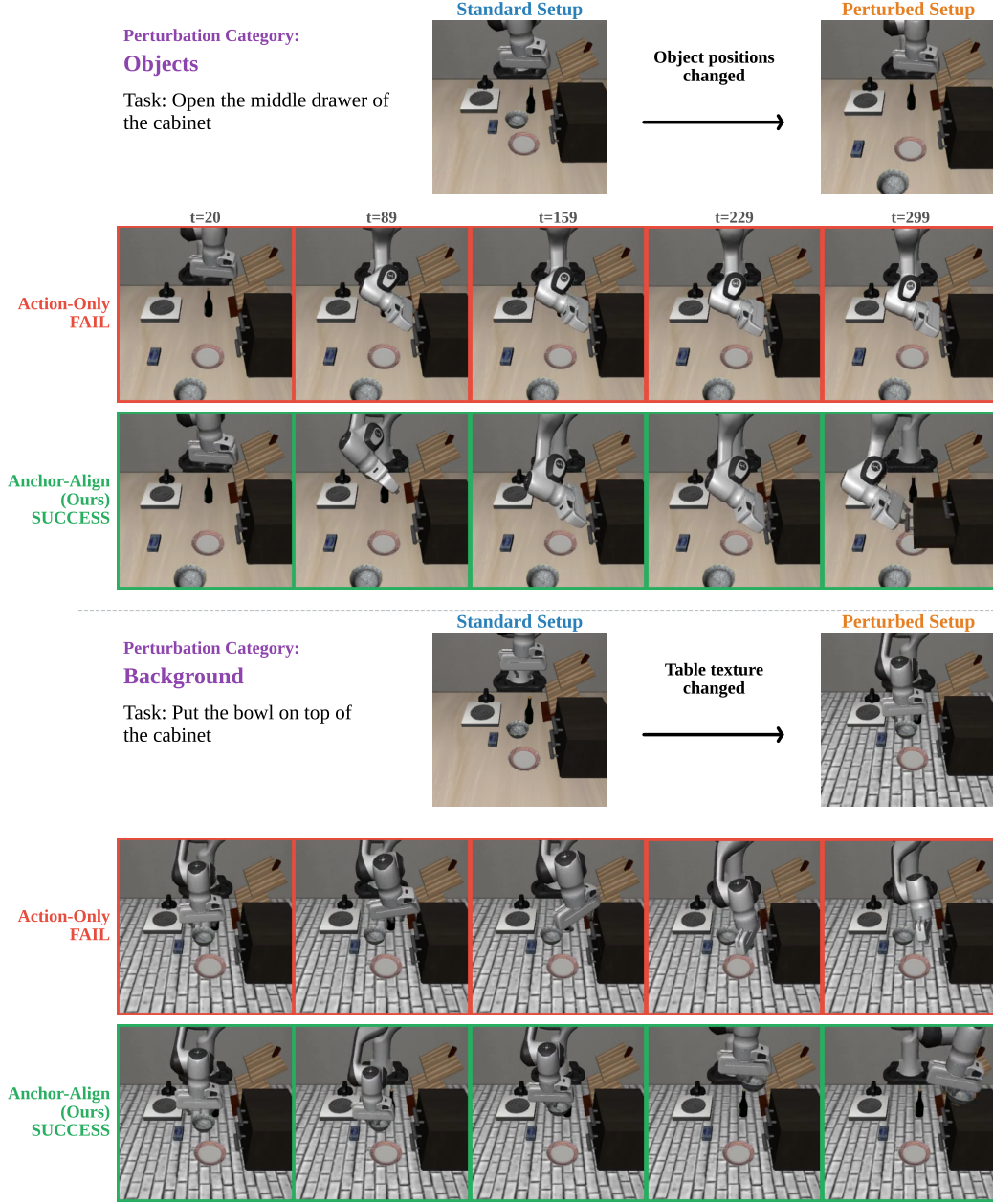


Figure 22: **LIBERO-Goal Plus: Objects and Background perturbations.** *Top:* Objects perturbation on “Open the middle drawer of the cabinet”—the cabinet, plate, and surrounding distractors are rearranged. Standard BC (red) hovers above the wrong region without contacting the handle; Anchor-Align VLA (green) locates the middle drawer and opens it. *Bottom:* Background perturbation on “Put the bowl on top of the cabinet”—the wood floor is swapped for a stone-tile surface. Standard BC drifts away from the cabinet; Anchor-Align VLA grasps the bowl and deposits it on top of the cabinet.

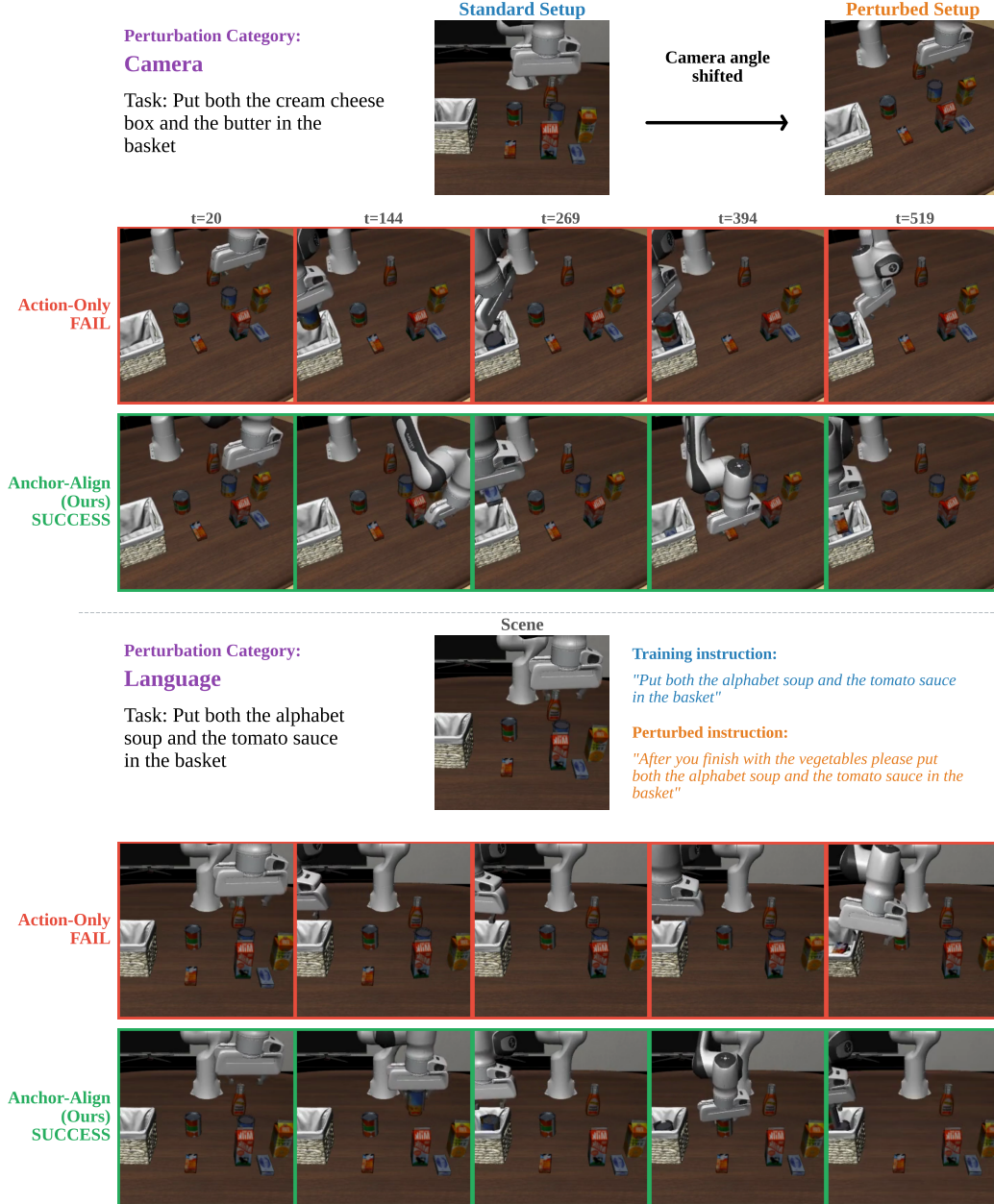


Figure 23: **LIBERO-Long Plus: Camera and Language perturbations.** *Top:* Camera perturbation on the long-horizon task “Put both the cream cheese box and the butter in the basket”: the camera viewpoint is shifted. Standard BC (red) picks up the incorrect object and places it in the basket, showcasing its reliance on memorized scene-to-action mappings that break under the viewpoint shift; Anchor-Align VLA (green) sequentially picks both targets and drops them in the basket. *Bottom:* Language perturbation on “Put both the alphabet soup and the tomato sauce in the basket”: the instruction is rephrased to “After you finish with the vegetables please put both the alphabet soup and the tomato sauce in the basket”. Standard BC fails to initiate the correct grasps; Anchor-Align VLA follows the rephrased instruction to completion.

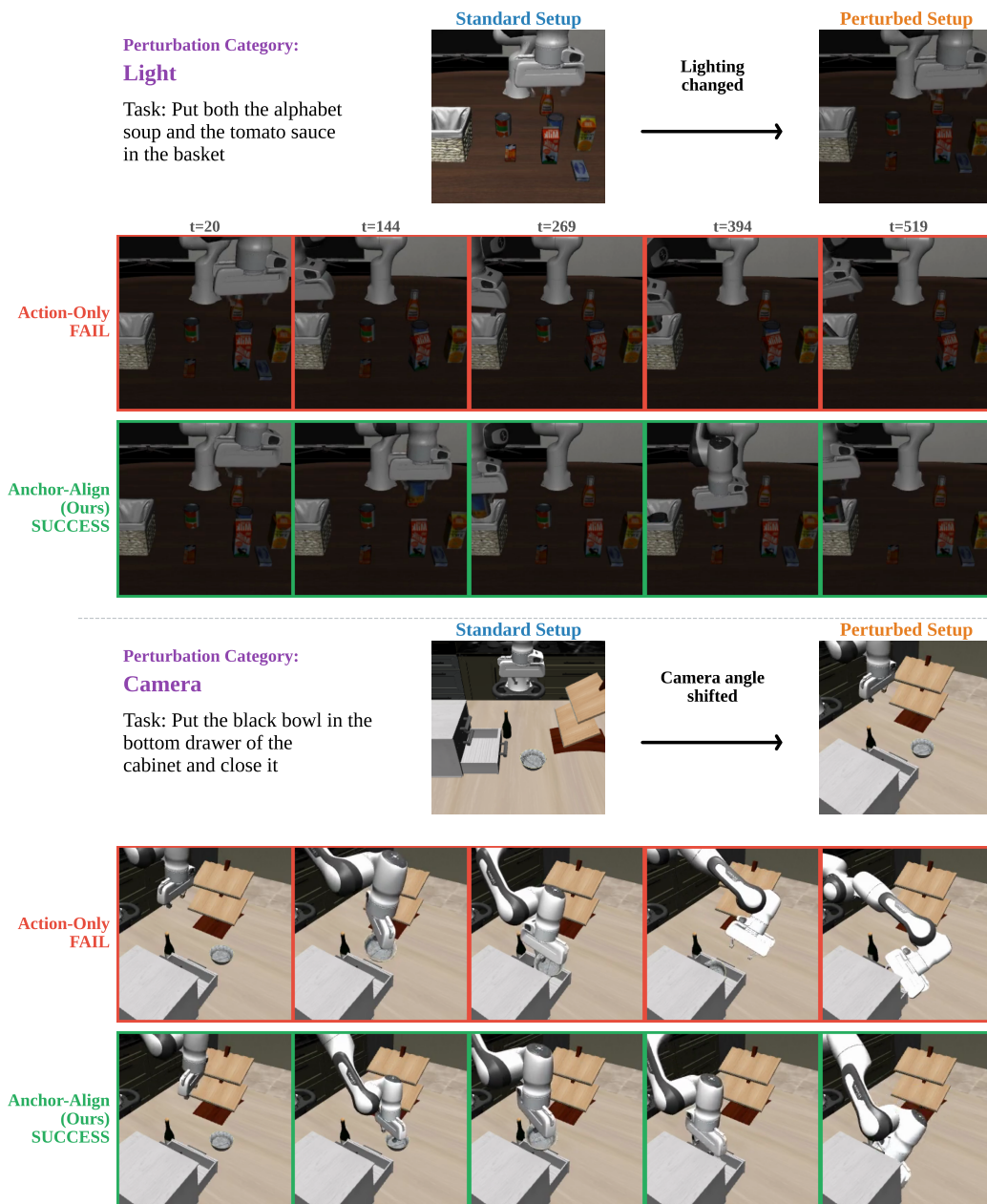


Figure 24: **LIBERO-Long Plus: Lighting and Camera perturbations.** *Top:* Lighting perturbation on “Put both the alphabet soup and the tomato sauce in the basket”: the scene is darkened. Standard BC (red) grasps an incorrect object, places it in the basket, and then idles without ever reaching for the second target; Anchor-Align VLA (green) deposits both items in the basket. *Bottom:* Camera perturbation on the long-horizon task “Put the black bowl in the bottom drawer of the cabinet and close it”: the camera viewpoint is shifted. Standard BC pulls the wrong drawer and never completes the close step; Anchor-Align VLA opens the bottom drawer, places the black bowl inside, and closes it.

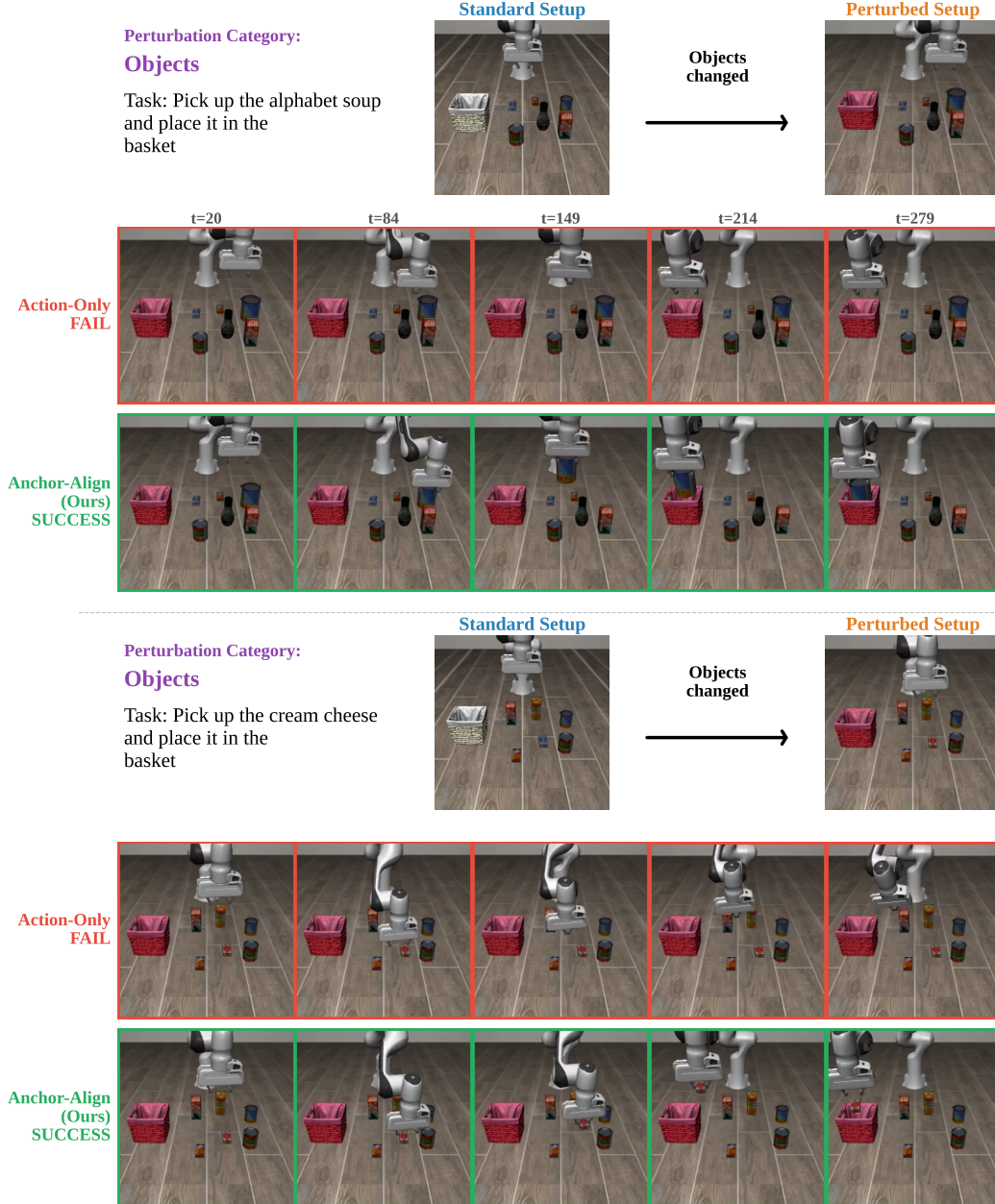


Figure 25: **LIBERO-PRO Object Swap**. Both rows use the same object-swap regime: the basket is replaced with a pink wicker variant and the surrounding canned items are exchanged for novel objects unseen during training. *Top*: “Pick up the alphabet soup and place it in the basket”: Standard BC (red) hovers over an incorrect target and then moves to the basket without grasping any object, whereas Anchor-Align VLA (green) grounds to the alphabet-soup can and deposits it in the basket. *Bottom*: “Pick up the cream cheese and place it in the basket”: Standard BC correctly identifies the target object but fails to grasp it, completing its trajectory with an empty gripper, whereas Anchor-Align VLA selects the cream-cheese box and places it correctly.